10-405/10-605:
Machine Learning with Large Datasets

Spring 2024

Distributed Systems, Map-Reduce, Spark

Announcements

- **HW1**: due *Monday January 29th* at 11:59pm ET
  - please start early!
  - use Piazza, office hours, recitation for questions
    - but don’t provide direct answers on these forums

- **Questions about Spark/Databricks?**
  - Watch recitation from Friday

PERFORMING THE ML PIPELINE AT SCALE

Key course topics

Data preparation
- Data cleaning
- Data summarization
- Visualization
- Dimensionality reduction

Training
- Distributed ML
- Large-scale optimization
- Scalable deep learning
- Efficient data structures
- Hyperparameter tuning

Inference
- Hardware considerations
- Low-latency inference (compression, pruning, distillation)

Infrastructure / Frameworks
- Apache Spark
- TensorFlow
- AWS / Google Cloud / Azure

Advanced topics
- AutoML for Diverse Tasks
- Scaling Laws for FMs
- Training LLMs @ Mosaic
- RL with large data

Outline

1. Big Data: Challenges and Opportunities
2. MapReduce / Hadoop
3. Datacenter Communication Hierarchy
4. Spark and ML
5. HW1 preview

MOTIVATION

Where is all of this data coming from?

SOURCES OF BIG DATA

Online data

Lots of user-generated data online

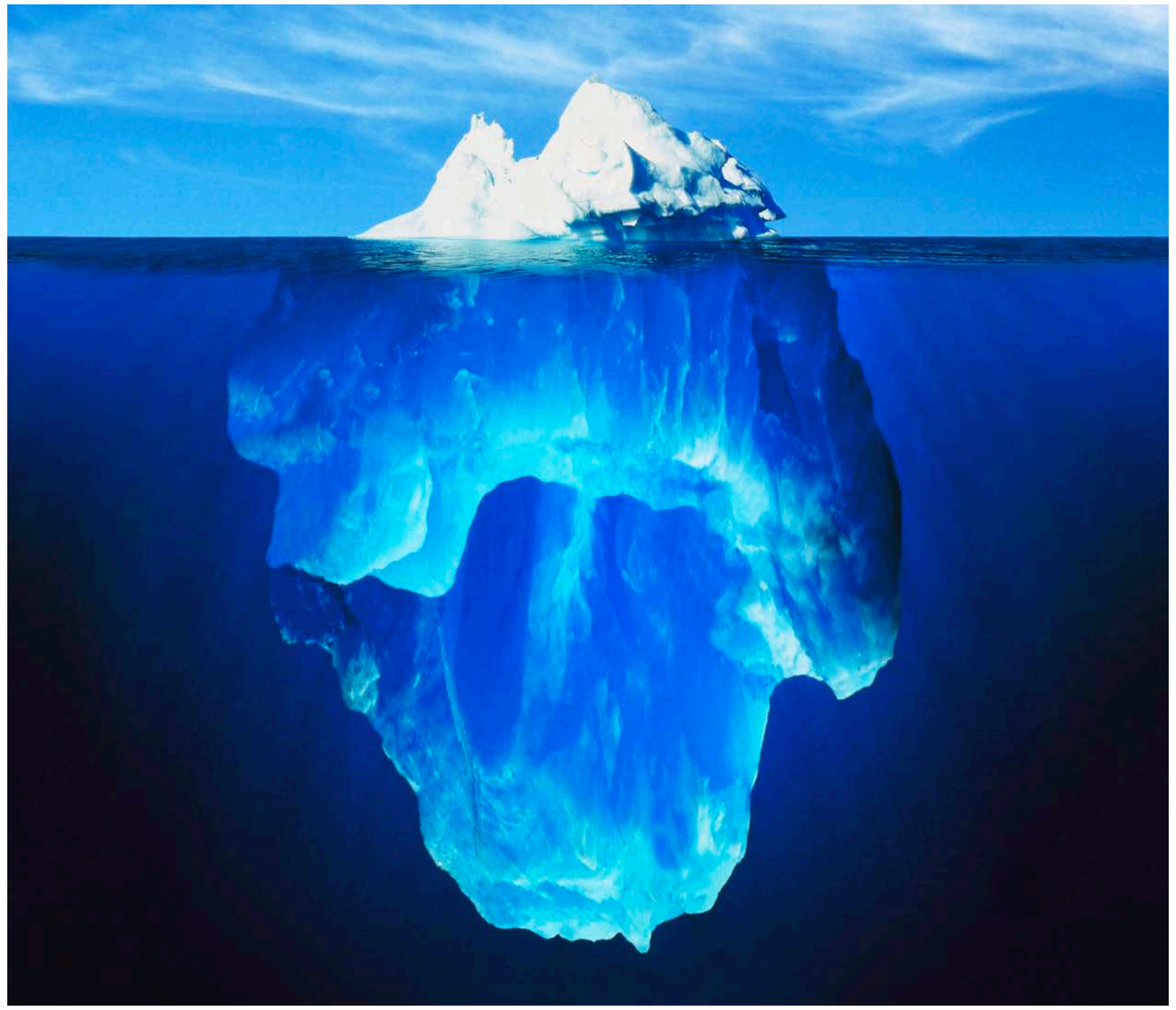
- instagram posts
- tweets
- youtube videos
- yelp reviews
- facebook posts
- stack overflow
- reddit

SOURCES OF BIG DATA

Online data

You may be generating data online without realizing it

- clicks, likes, retweets
- frequency/duration of page visits
- page highlights
- billing event
- error logs
- videos: fast forward, pause, rewind, …



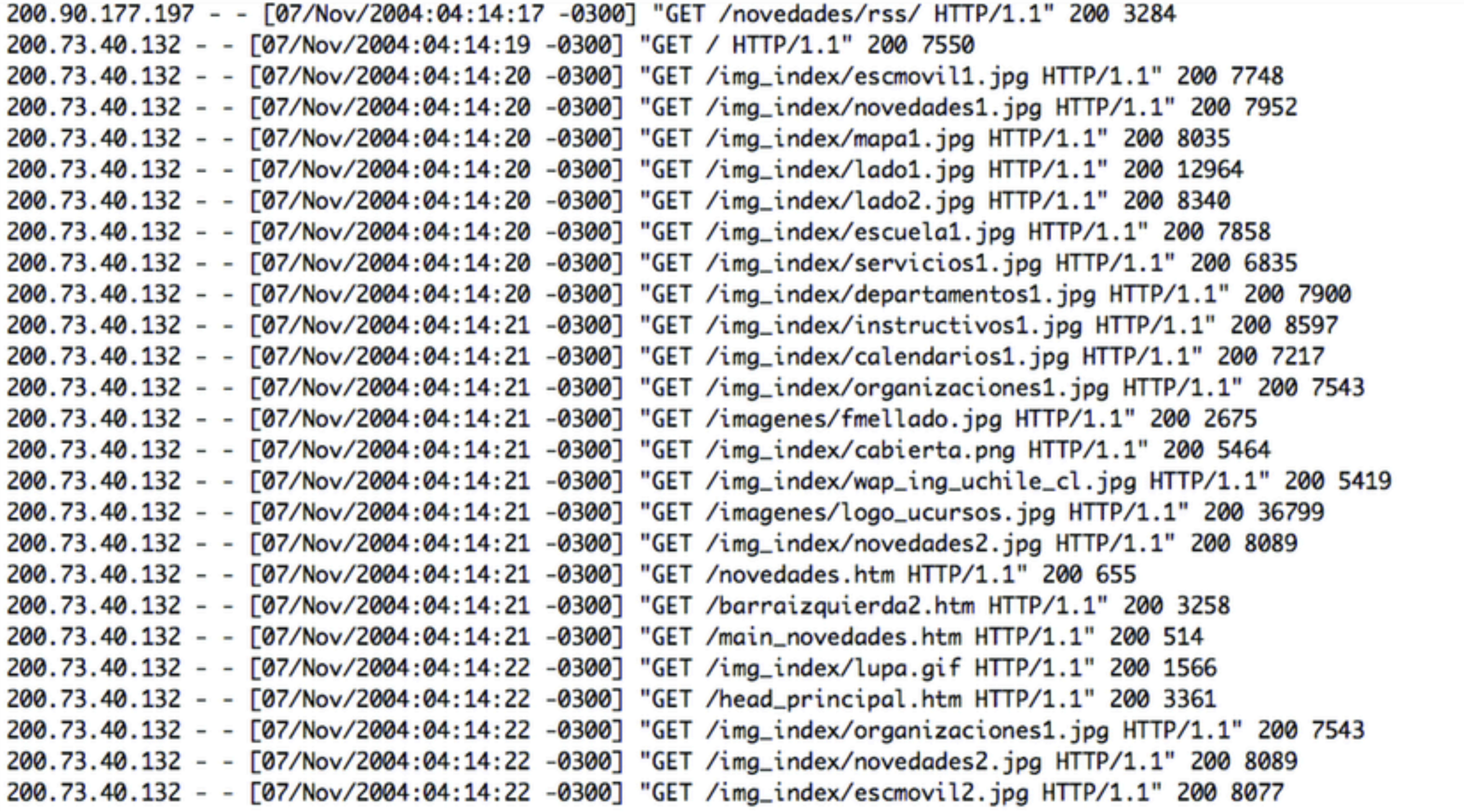
SOURCES OF BIG DATA

Graph data

- social networks
- google’s knowledge graph
- telecommunication networks
- road networks
- disease transmission

SOURCES OF BIG DATA

Log data

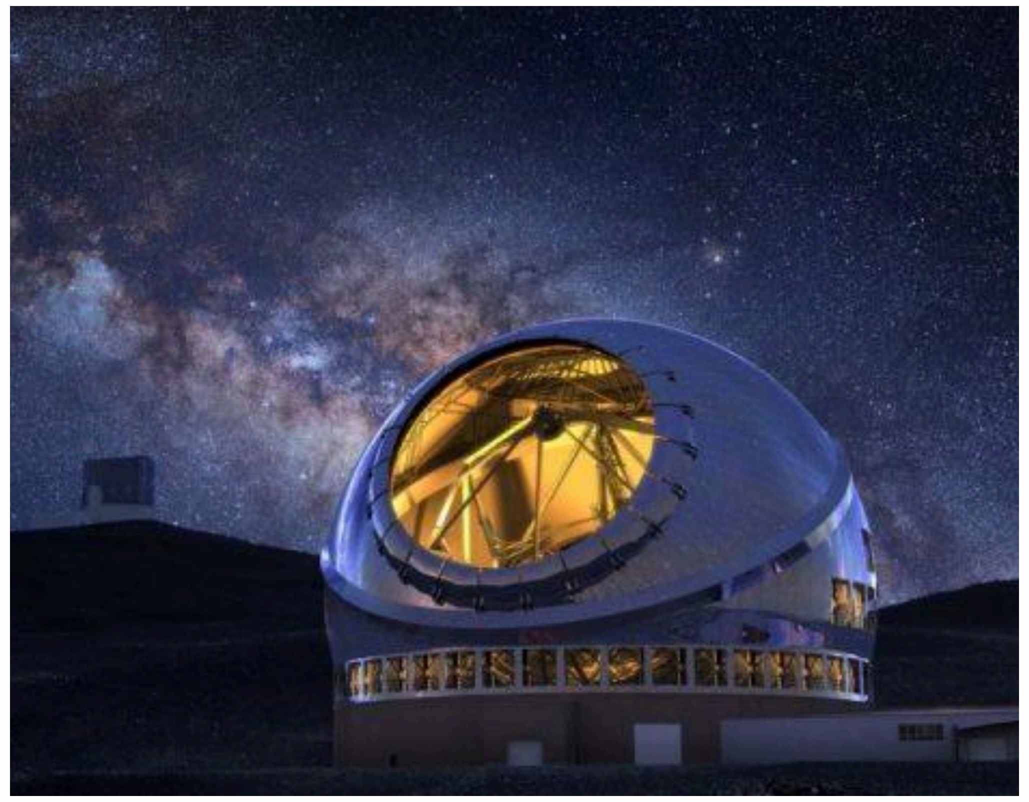


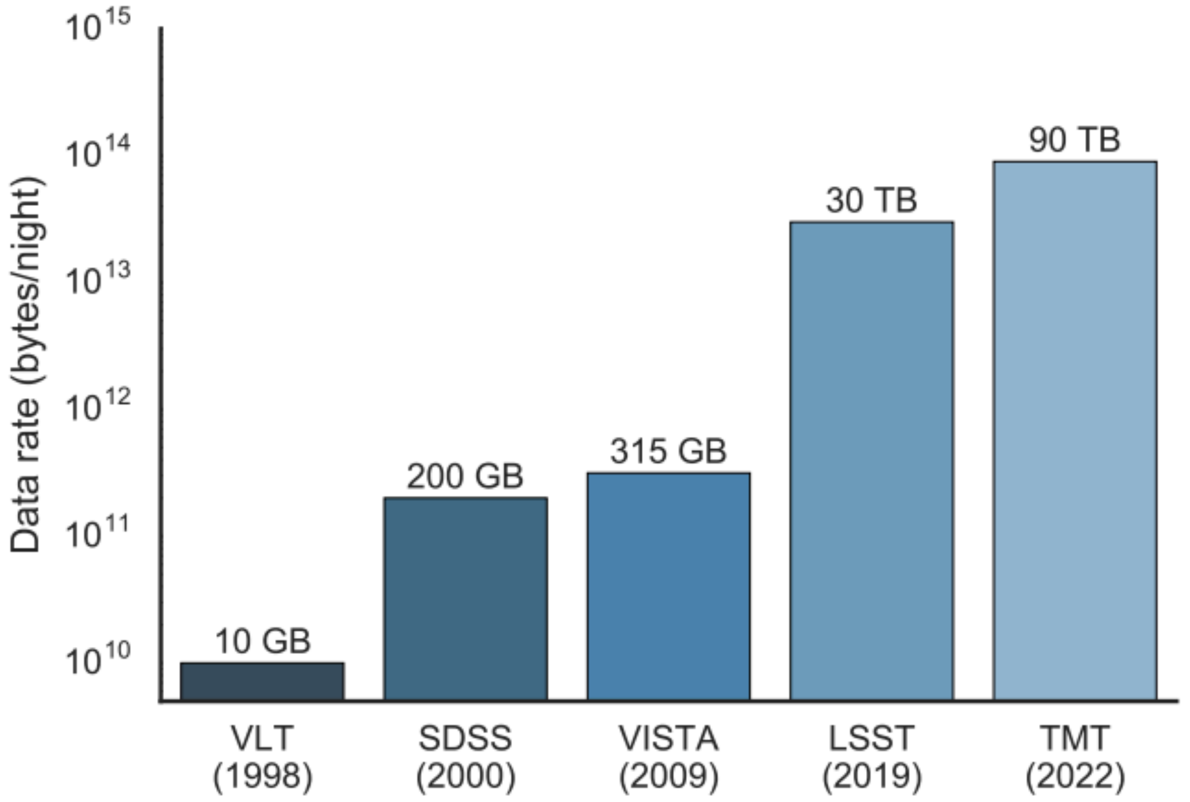
```
200.90.177.197 - - [07/Nov/2004:04:14:17 -0300] "GET /novedades/rss/ HTTP/1.1" 200 3284
200.73.40.132 - - [07/Nov/2004:04:14:19 -0300] "GET / HTTP/1.1" 200 7550
200.73.40.132 - - [07/Nov/2004:04:14:20 -0300] "GET /img_index/escmovil1.jpg HTTP/1.1" 200 7748
200.73.40.132 - - [07/Nov/2004:04:14:20 -0300] "GET /img_index/novedades1.jpg HTTP/1.1" 200 7952
200.73.40.132 - - [07/Nov/2004:04:14:20 -0300] "GET /img_index/mapa1.jpg HTTP/1.1" 200 8035
200.73.40.132 - - [07/Nov/2004:04:14:20 -0300] "GET /img_index/lado1.jpg HTTP/1.1" 200 12964
200.73.40.132 - - [07/Nov/2004:04:14:20 -0300] "GET /img_index/lado2.jpg HTTP/1.1" 200 8340
200.73.40.132 - - [07/Nov/2004:04:14:20 -0300] "GET /img_index/escuela1.jpg HTTP/1.1" 200 7858
200.73.40.132 - - [07/Nov/2004:04:14:20 -0300] "GET /img_index/servicios1.jpg HTTP/1.1" 200 6835
200.73.40.132 - - [07/Nov/2004:04:14:20 -0300] "GET /img_index/departamentos1.jpg HTTP/1.1" 200 7900
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /img_index/instructivos1.jpg HTTP/1.1" 200 8597
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /img_index/calendarios1.jpg HTTP/1.1" 200 7217
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /img_index/organizaciones1.jpg HTTP/1.1" 200 7543
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /imagenes/fmellado.jpg HTTP/1.1" 200 2675
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /img_index/cabierta.png HTTP/1.1" 200 5464
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /img_index/wap_ing_uchile_cl.jpg HTTP/1.1" 200 5419
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /imagenes/logo_ucursos.jpg HTTP/1.1" 200 36799
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /img_index/novedades2.jpg HTTP/1.1" 200 8089
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /novedades.htm HTTP/1.1" 200 655
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /barraizquierda2.htm HTTP/1.1" 200 3258
200.73.40.132 - - [07/Nov/2004:04:14:21 -0300] "GET /main_novedades.htm HTTP/1.1" 200 514
200.73.40.132 - - [07/Nov/2004:04:14:22 -0300] "GET /img_index/lupa.gif HTTP/1.1" 200 1566
200.73.40.132 - - [07/Nov/2004:04:14:22 -0300] "GET /head_principal.htm HTTP/1.1" 200 3361
200.73.40.132 - - [07/Nov/2004:04:14:22 -0300] "GET /img_index/organizaciones1.jpg HTTP/1.1" 200 7543
200.73.40.132 - - [07/Nov/2004:04:14:22 -0300] "GET /img_index/novedades2.jpg HTTP/1.1" 200 8089
200.73.40.132 - - [07/Nov/2004:04:14:22 -0300] "GET /img_index/escmovil2.jpg HTTP/1.1" 200 8077
```

SOURCES OF BIG DATA

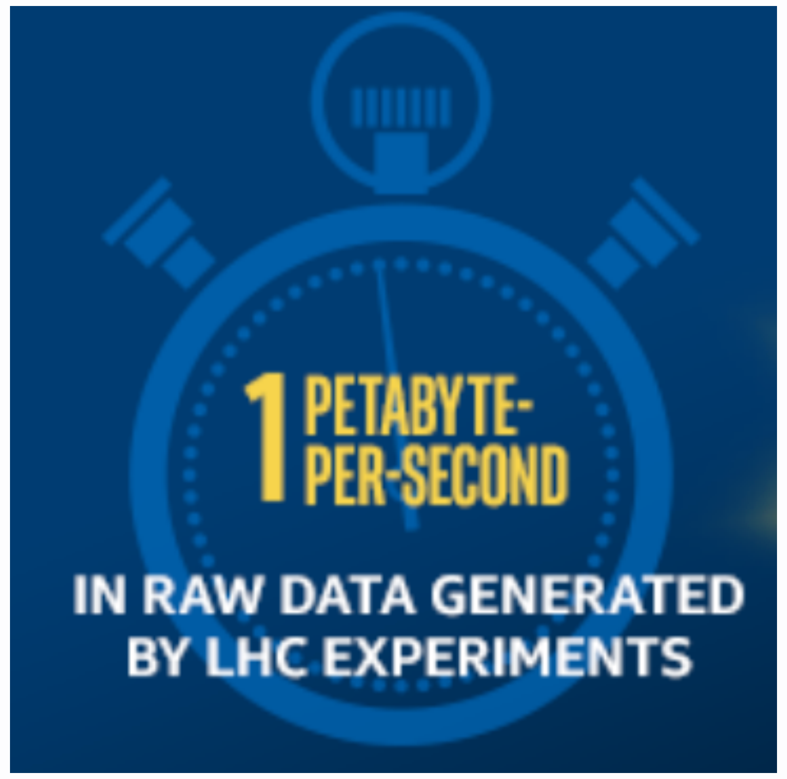
Scientific data

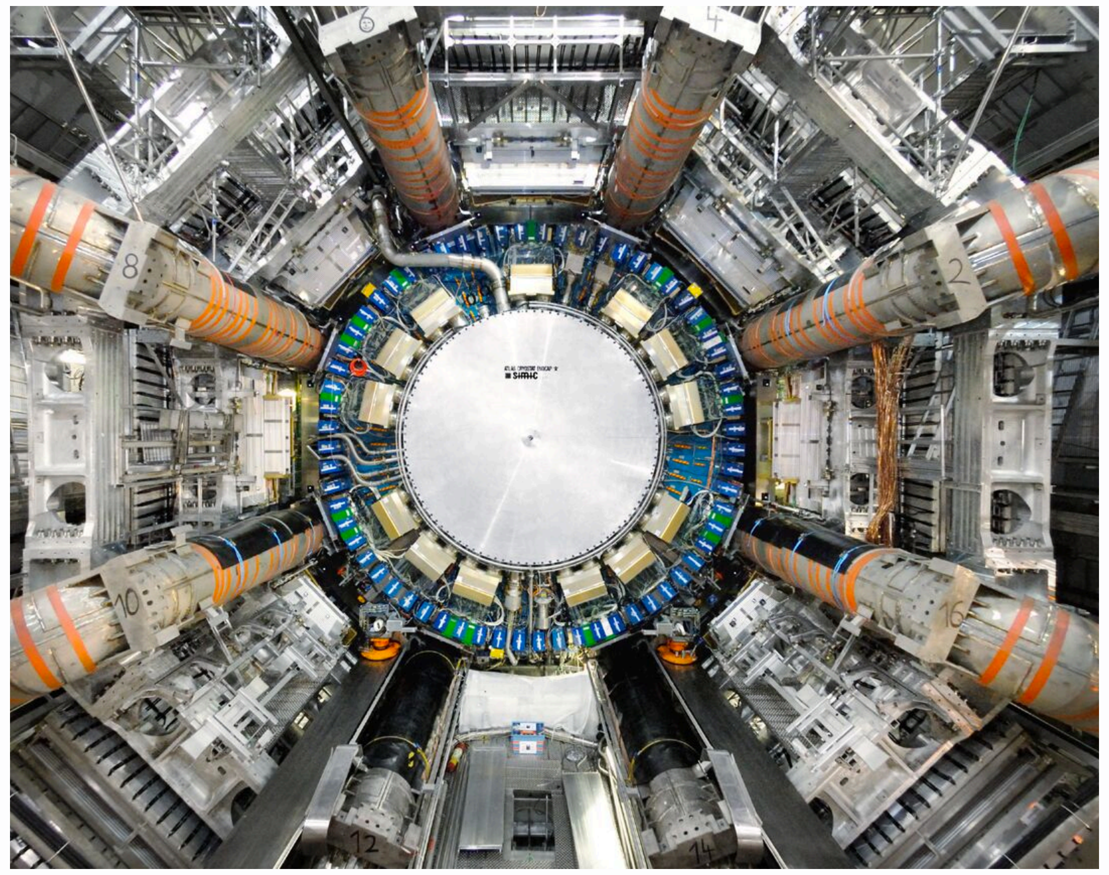
astronomy



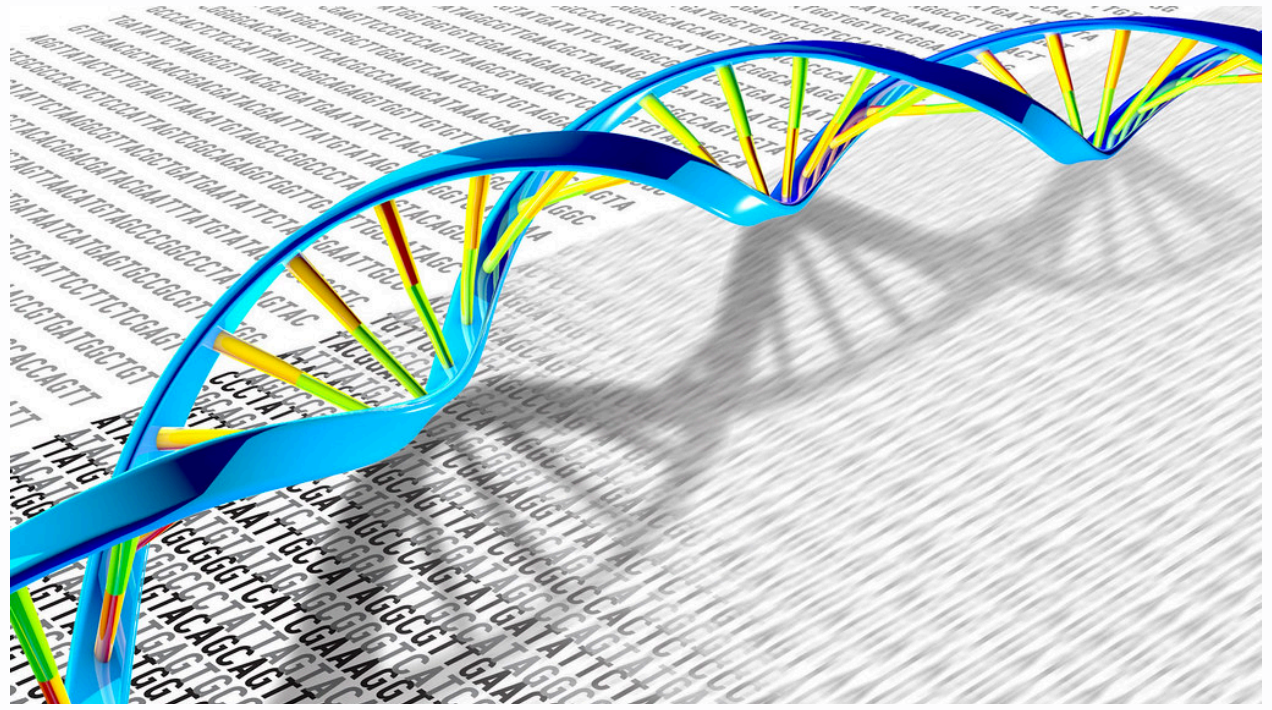


physics

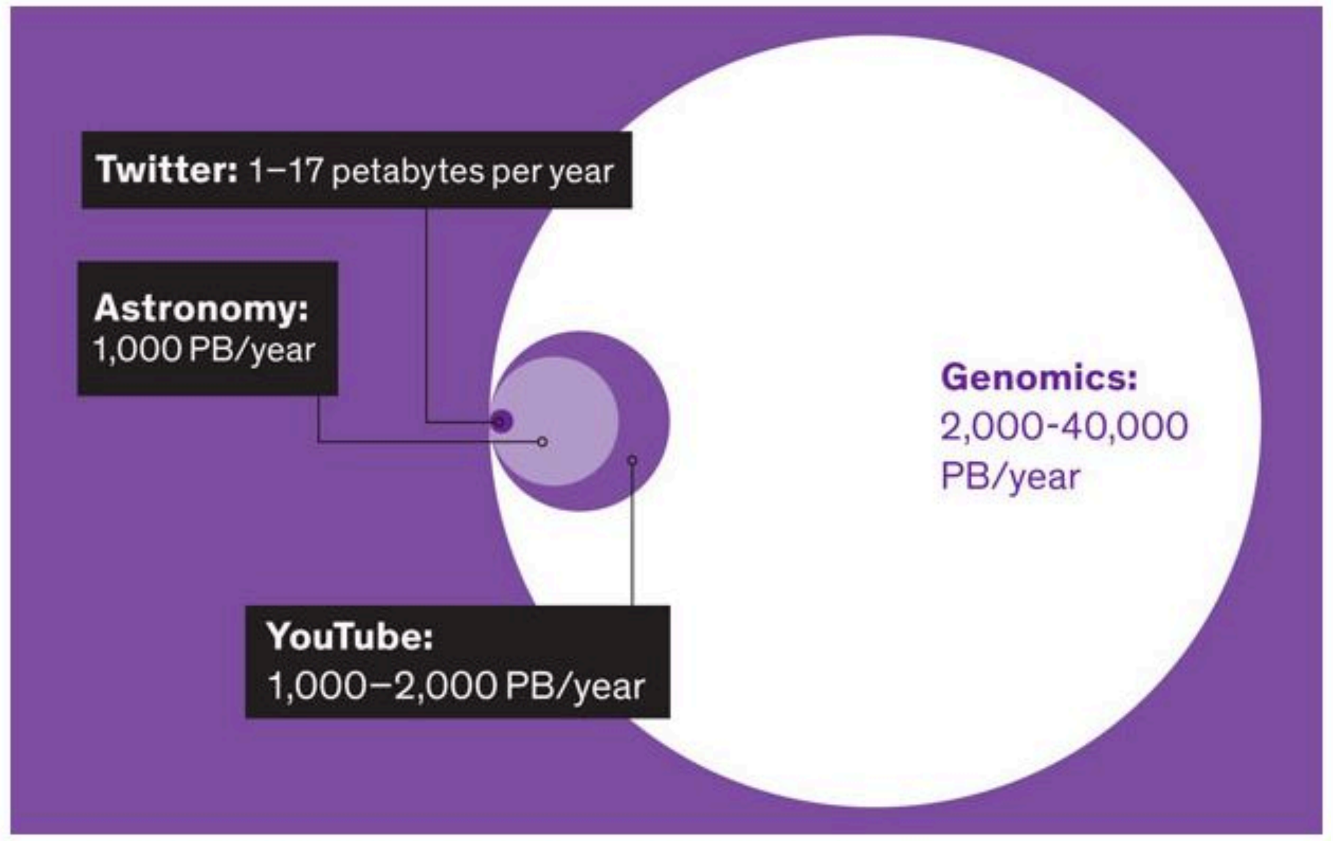

1 PETABYTE-PER-SECOND
IN RAW DATA GENERATED BY LHC EXPERIMENTS



genomics



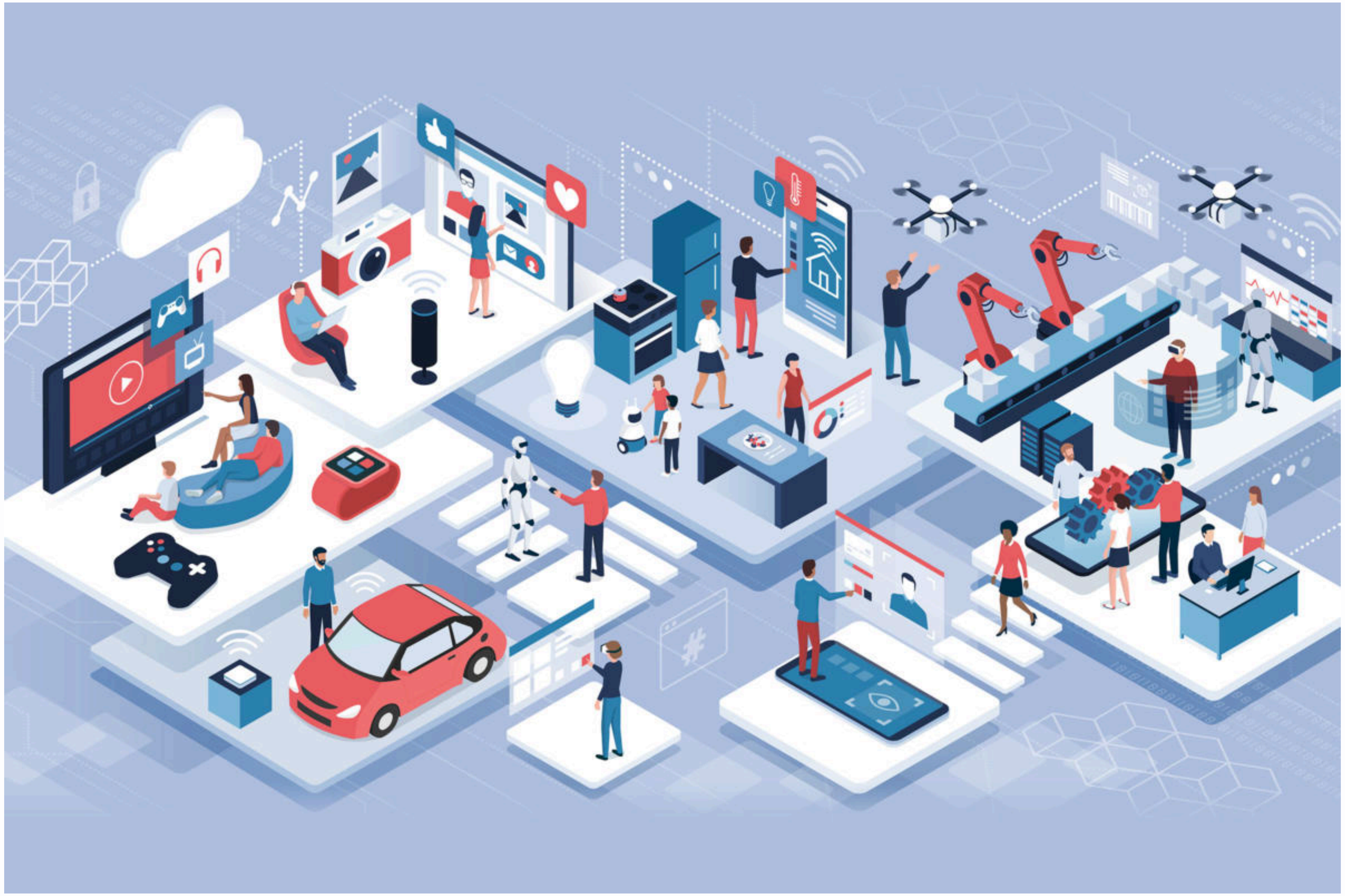
Projected annual storage in 2025



Source: "Big Data: Astronomical or Genomical?" PLoS Biology, 7 July 2015.

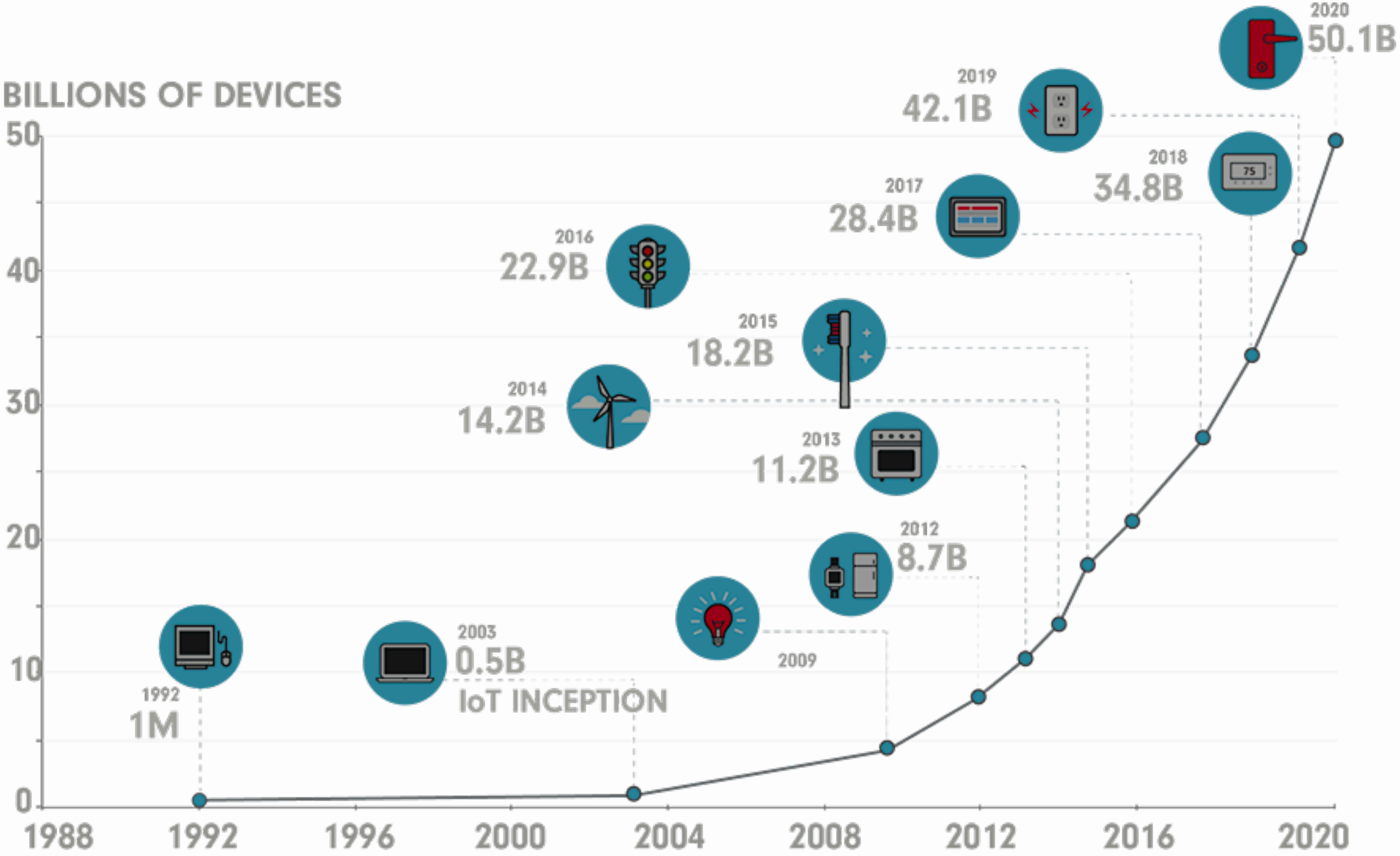
SOURCES OF BIG DATA

Internet of Things



SOURCES OF BIG DATA

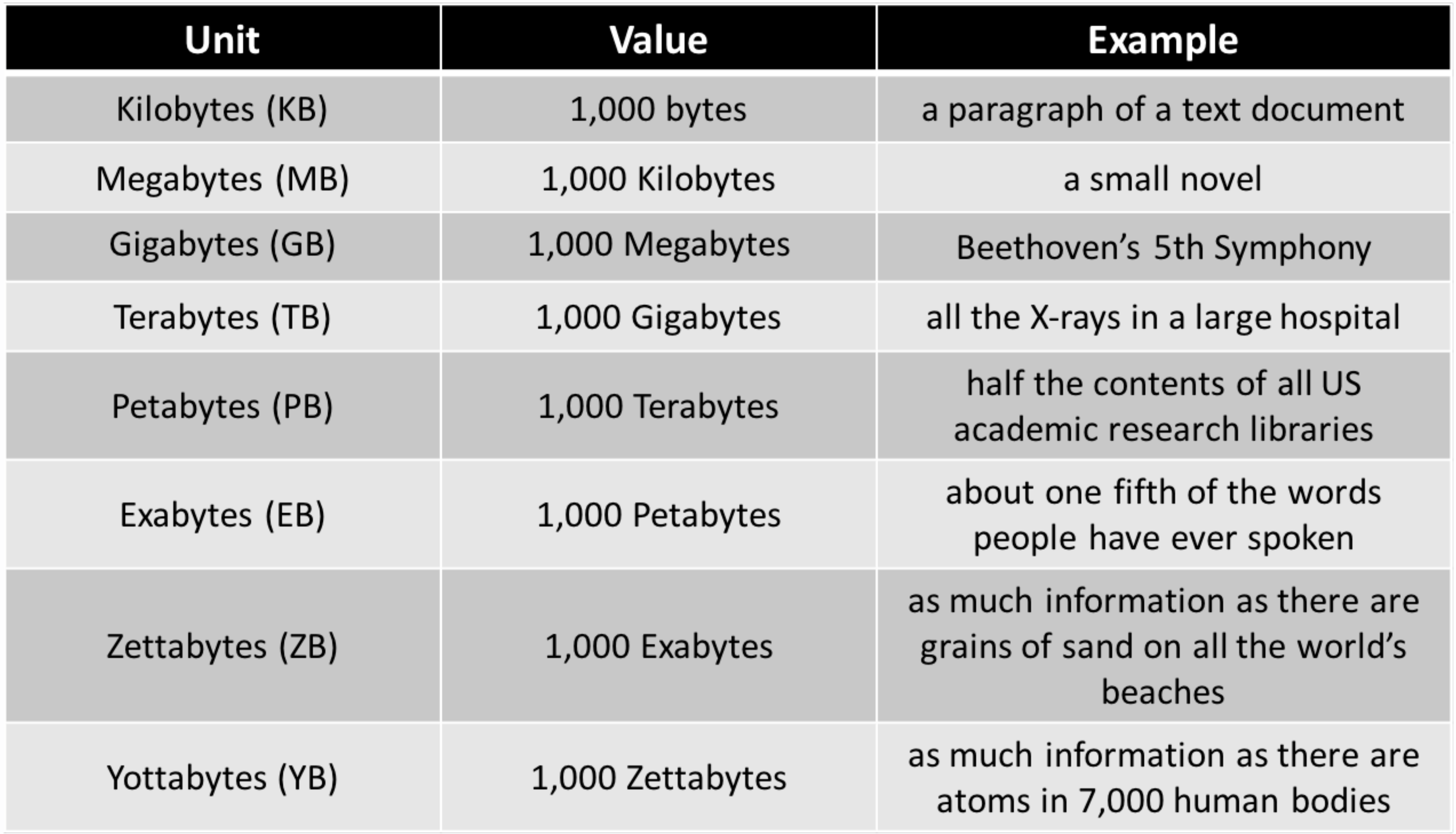
Internet of Things



Source: Cisco

ASIDE

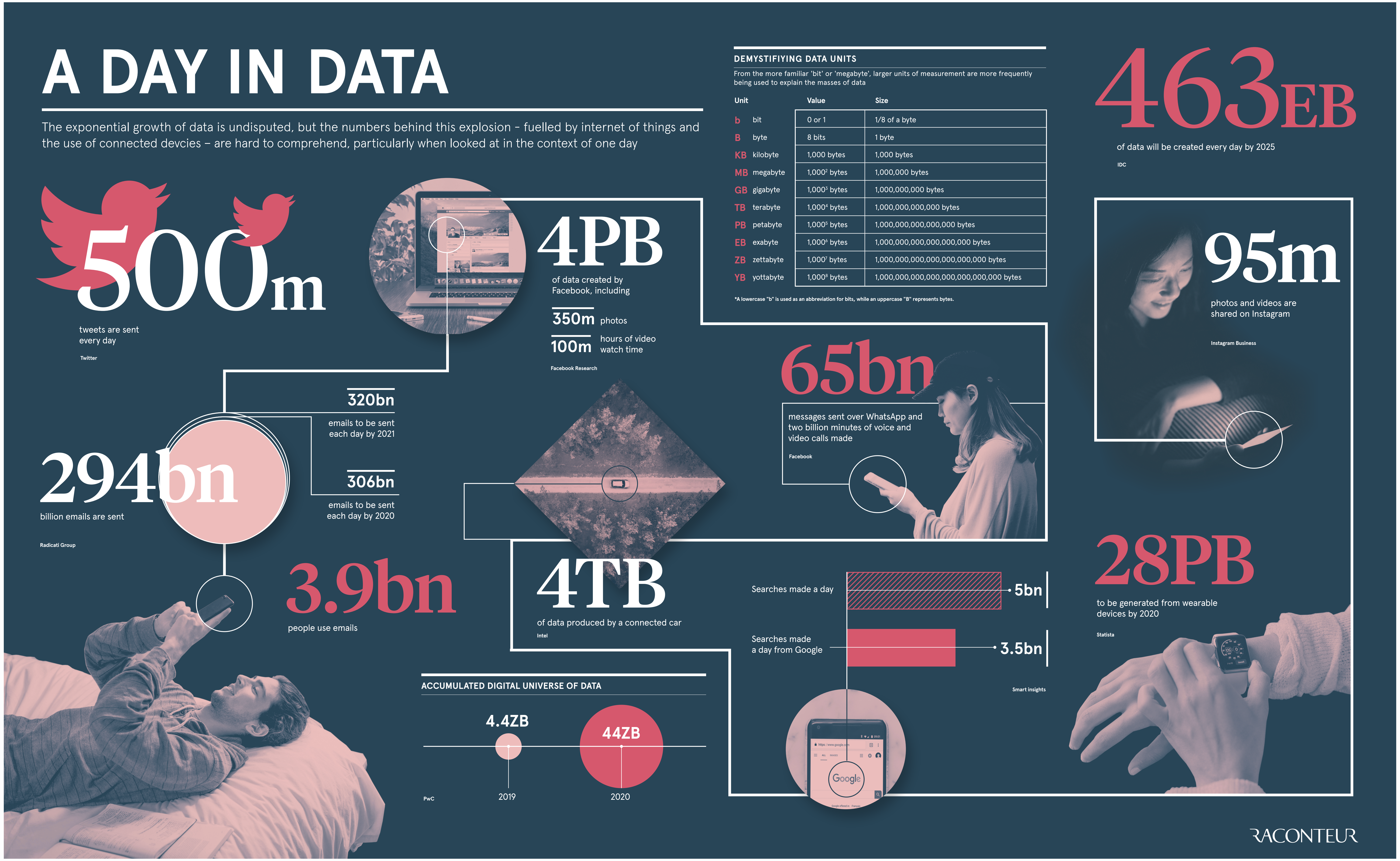
Data units



Unit	Value	Example
Kilobytes (KB)	1,000 bytes	a paragraph of a text document
Megabytes (MB)	1,000 Kilobytes	a small novel
Gigabytes (GB)	1,000 Megabytes	Beethoven's 5th Symphony
Terabytes (TB)	1,000 Gigabytes	all the X-rays in a large hospital
Petabytes (PB)	1,000 Terabytes	half the contents of all US academic research libraries
Exabytes (EB)	1,000 Petabytes	about one fifth of the words people have ever spoken
Zettabytes (ZB)	1,000 Exabytes	as much information as there are grains of sand on all the world's beaches
Yottabytes (YB)	1,000 Zettabytes	as much information as there are atoms in 7,000 human bodies

SUMMARY

LOTS OF DATA



A DAY IN DATA

The exponential growth of data is undisputed, but the numbers behind this explosion - fuelled by internet of things and the use of connected devcies – are hard to comprehend, particularly when looked at in the context of one day

500m
tweets are sent every day
Twitter

4PB
of data created by Facebook, including
350m photos
100m hours of video watch time
Facebook Research

294bn
billion emails are sent
Radicati Group

320bn
emails to be sent each day by 2021

306bn
emails to be sent each day by 2020

3.9bn
people use emails

4TB
of data produced by a connected car
Intel

65bn
messages sent over WhatsApp and two billion minutes of voice and video calls made
Facebook

DEMYSTIFIYING DATA UNITS

From the more familiar 'bit' or 'megabyte', larger units of measurement are more frequently being used to explain the masses of data

Unit		Value	Size
b	bit	0 or 1	1/8 of a byte
B	byte	8 bits	1 byte
KB	kilobyte	1,000 bytes	1,000 bytes
MB	megabyte	$1{,}000^2$ bytes	1,000,000 bytes
GB	gigabyte	$1{,}000^3$ bytes	1,000,000,000 bytes
TB	terabyte	$1{,}000^4$ bytes	1,000,000,000,000 bytes
PB	petabyte	$1{,}000^5$ bytes	1,000,000,000,000,000 bytes
EB	exabyte	$1{,}000^6$ bytes	1,000,000,000,000,000,000 bytes
ZB	zettabyte	$1{,}000^7$ bytes	1,000,000,000,000,000,000,000 bytes
YB	yottabyte	$1{,}000^8$ bytes	1,000,000,000,000,000,000,000,000 bytes

*A lowercase "b" is used as an abbreviation for bits, while an uppercase "B" represents bytes.

463EB
of data will be created every day by 2025
IDC

95m
photos and videos are shared on Instagram
Instagram Business

28PB
to be generated from wearable devices by 2020
Statista

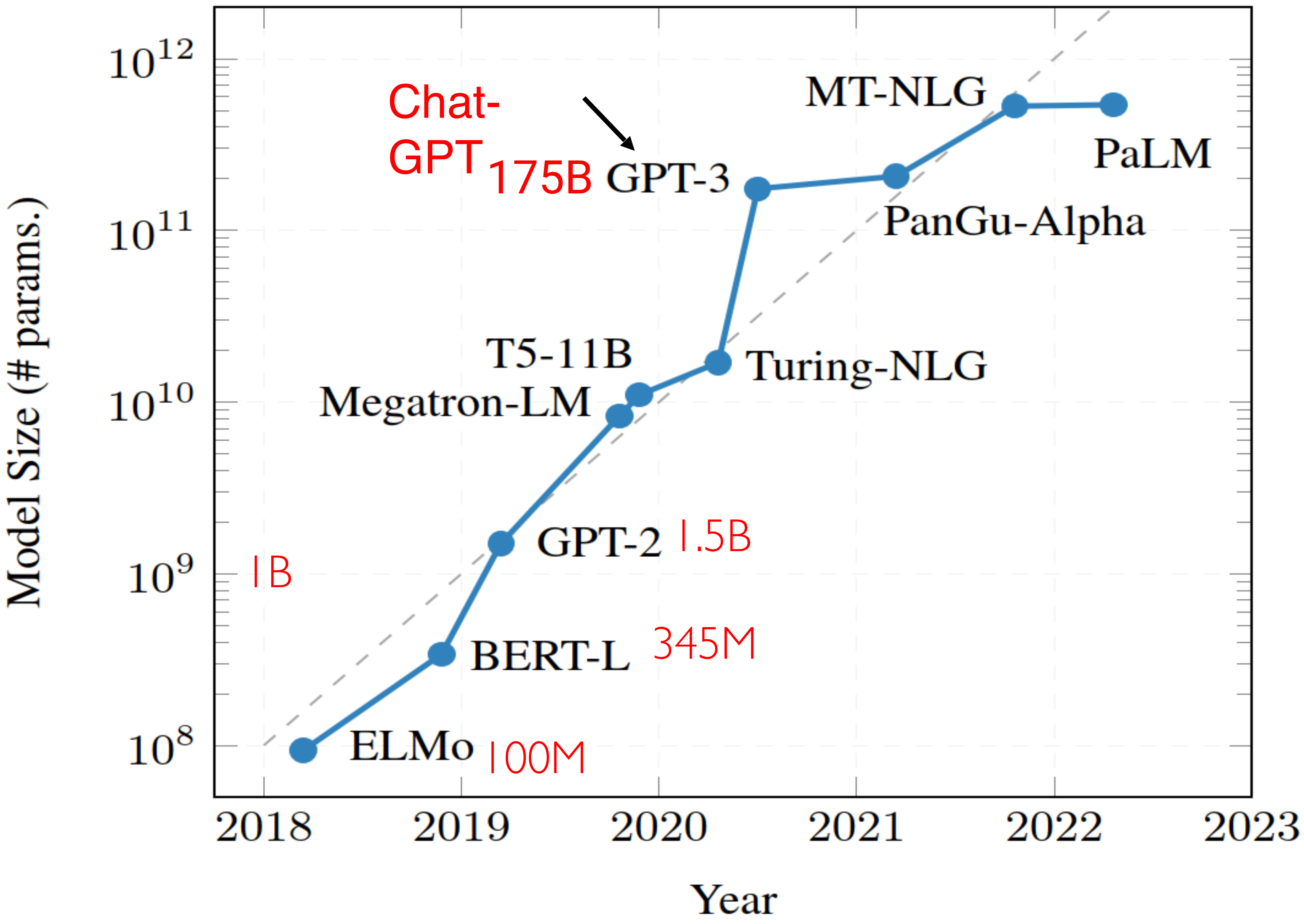
Searches made a day – 5bn
Searches made a day from Google – 3.5bn
Smart insights

ACCUMULATED DIGITAL UNIVERSE OF DATA

PwC

RACONTEUR

ML Models are getting larger and larger



- Some reports claim that GPT-4 has more than 1 trillion parameters
- BERT large: 345 million parameters
- BERT base: 110 million parameters.

HRMMM 🤔

What tools would we typically use to analyze interesting data?

Traditional Data Analysis Tools

Without having to worry about scale, typical tools include:

- Python frameworks (e.g., pandas, Scikit-learn, etc)
- R frameworks
- Good ole' shell commands (grep, awk, sed)

But all of these run on a single machine!

The Big Data Problem

Data is growing faster than computation speeds

Growing number and size of data sources:

👉 Mobile, web, scientific, even agricultural & manufacturing...

Storage is getting cheaper!

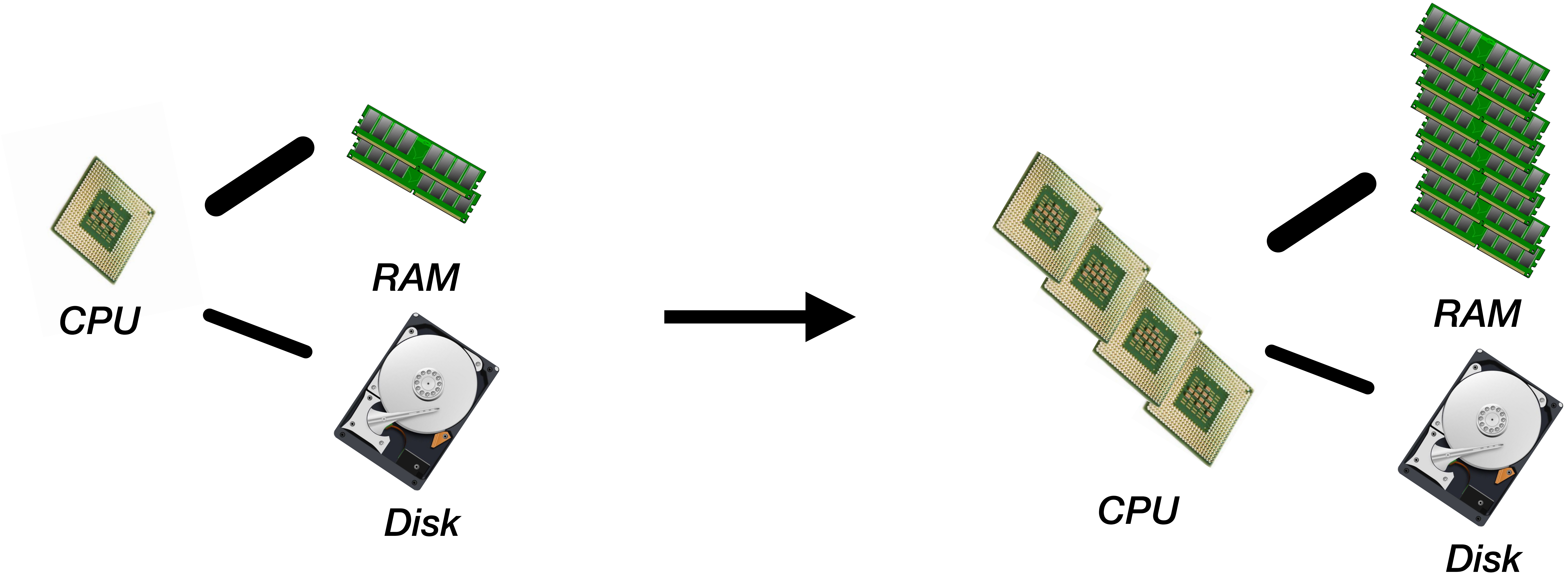
👉 Size doubling every 18 months

How to Handle Massive Data?

Need more hardware to store / process modern data

Scale-up (one big machine)

- Can be very fast for medium scale problems
- Expensive, specialized hardware
- Eventually hit a wall


CPU
RAM
Disk
CPU
RAM
Disk

We can’t process or even store all the data on a single machine 🤯

The Big Data Problem

Examples of the data that’s being created and that we would like to analyze...

Facebook’s daily logs: **60 TB**

1,000 genomes project: **200 TB**

Google web index: **10+ PB**

Cost of 1TB of disk: **~$35**

Time to read 1TB from disk: **3 hours**

Cloud Computing is an Enabler

Cloud computing is a big deal!

Provides access to low-cost computing and storage

Costs decreasing every year

aws

Google Cloud Platform

Microsoft Azure

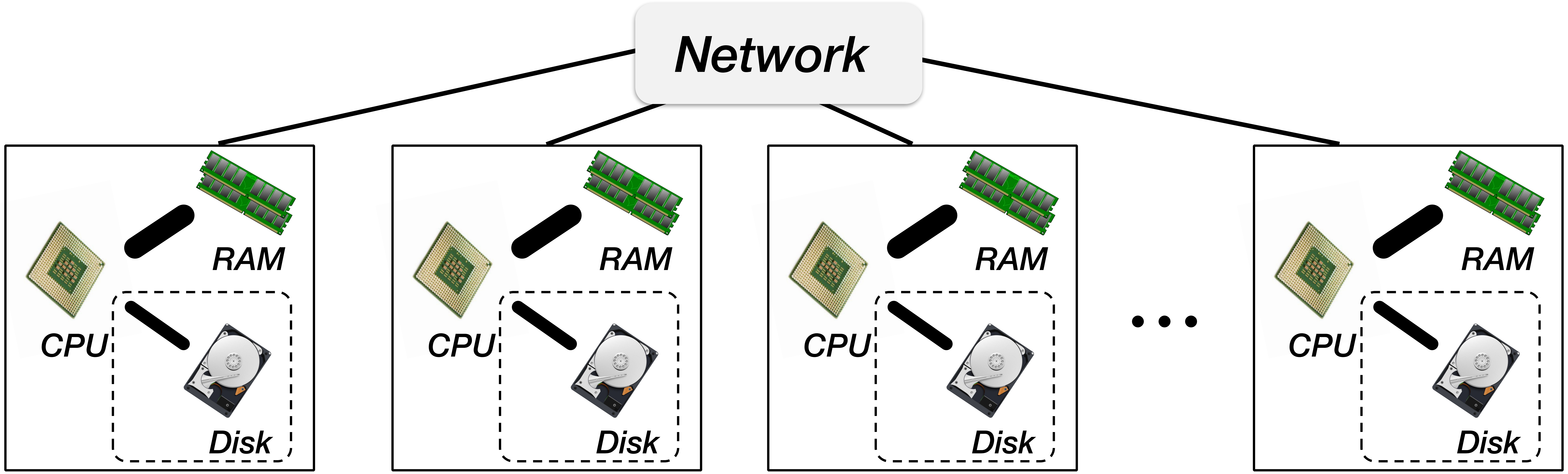
ORACLE Cloud

How to Handle Massive Data?

Need more hardware to store / process modern data

Scale-out (distributed, e.g., cloud-based)

- Commodity hardware, scales to massive problems
- Need to deal with network communication
- Added software complexity


Network
CPU
RAM
Disk
CPU
RAM
Disk
CPU
RAM
Disk
…
CPU
RAM
Disk

Google Datacenter

How do we program this thing!?

Google Datacenter

How do we program this thing!?

Disclaimer: Not everything is a “Big Data” problem!

E.g., Lots of practical ML still focuses on “small data”

E.g., Some deep learning takes the scale-up approach

Outline

1. Big Data: Challenges and Opportunities
2. MapReduce / Hadoop
3. Datacenter Communication Hierarchy
4. Spark and ML
5. HW1 preview

Hardware for Big Data

Image: Steve Jurvetson/Flickr

Consumer-grade hardware

  Not "gold plated"

Many desktop-like servers

  Easy to add capacity

  Cheaper per CPU/disk

But, requires complexity in software

Problems with Cheap Hardware

Facebook Datacenter (2014)

Failures, Google's numbers:
- 1-5% hard drives/year
- 0.2% DIMMs/year

Network speeds versus shared memory
- *Much* more latency
- Network slower than storage

Uneven performance

What's Hard About Cluster Computing?

How do we split work across machines?

Let's look at a simple task: word counting

How do you count the number of occurrences of each word in a document?

“I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?”

I: 3
am: 3
Sam: 3
do: 1
you: 1
like: 1
…

One Approach: Use a Hash Table

"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?"

{}

One Approach: Use a Hash Table

“I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?”

{I :**1**}

One Approach: Use a Hash Table

"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?"

{I: **1**,
am: **1**}

One Approach: Use a Hash Table

“I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?”

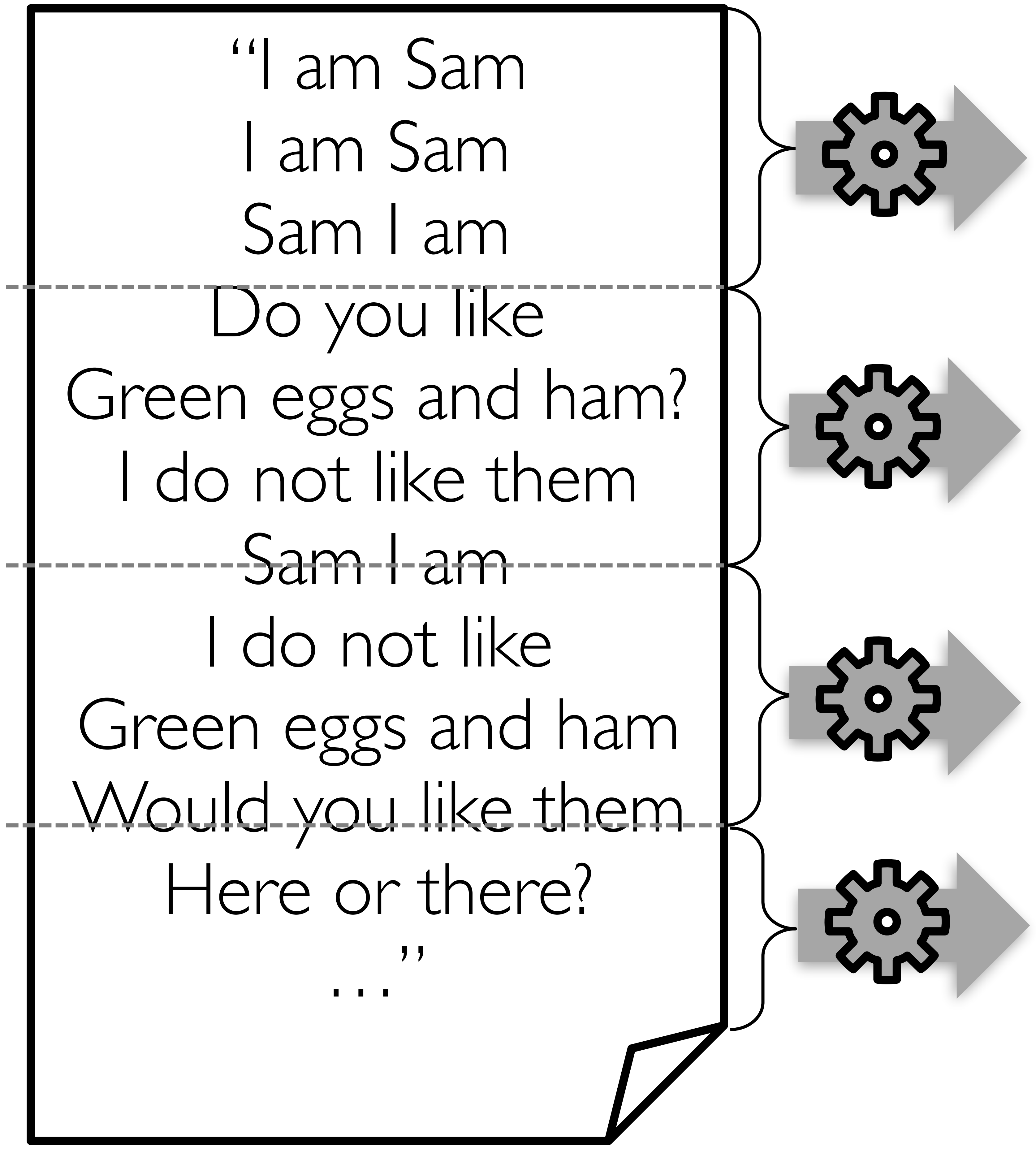
{I: **1**,
am: **1**,
Sam: **1**}

One Approach: Use a Hash Table

"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?"

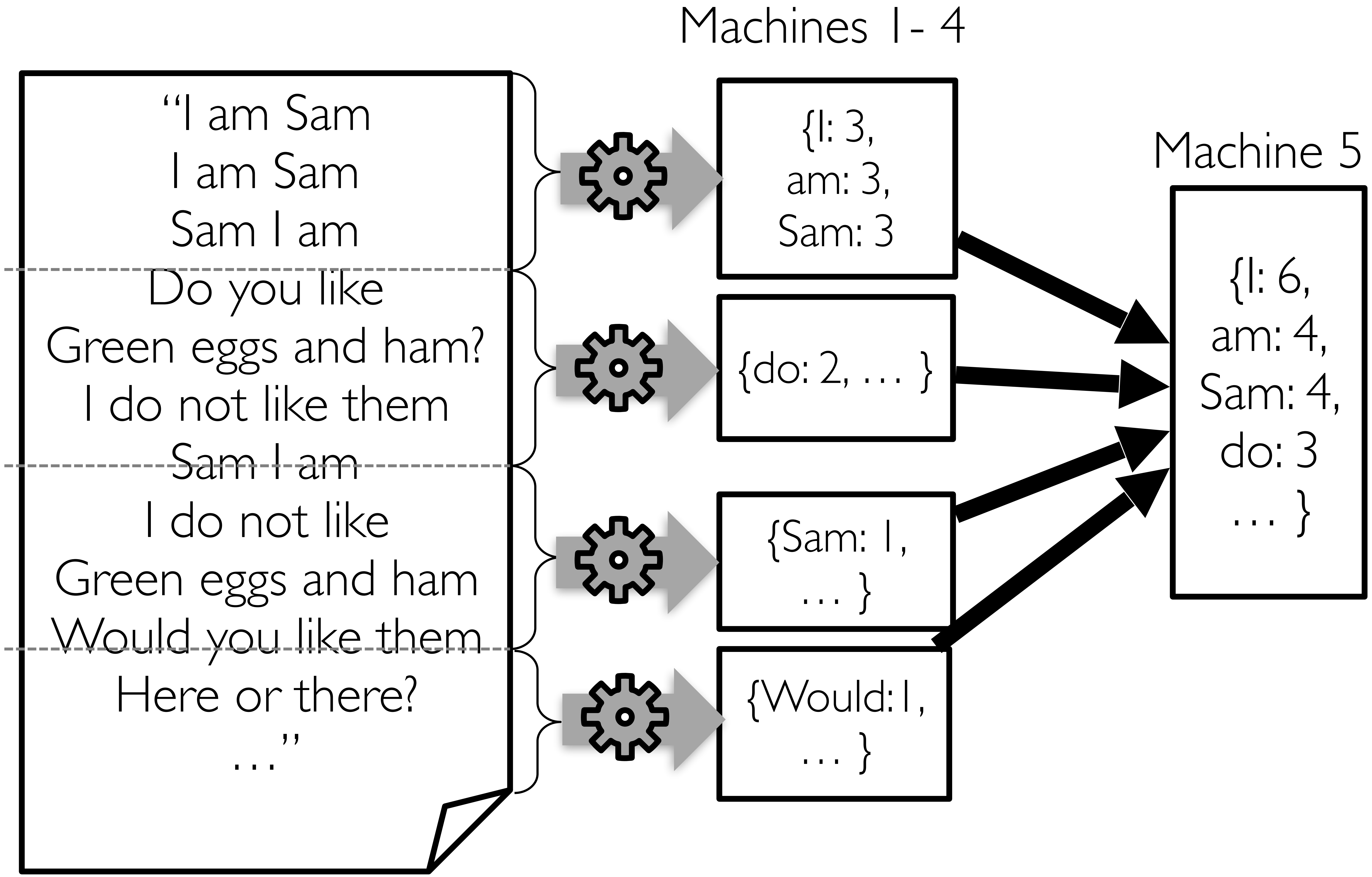
{I: 2,
am: 1,
Sam: 1}

What if the Document is Really Big?



"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"

What if the Document is Really Big?



Machines 1- 4

"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"

{I: 3,
am: 3,
Sam: 3

{do: 2, … }

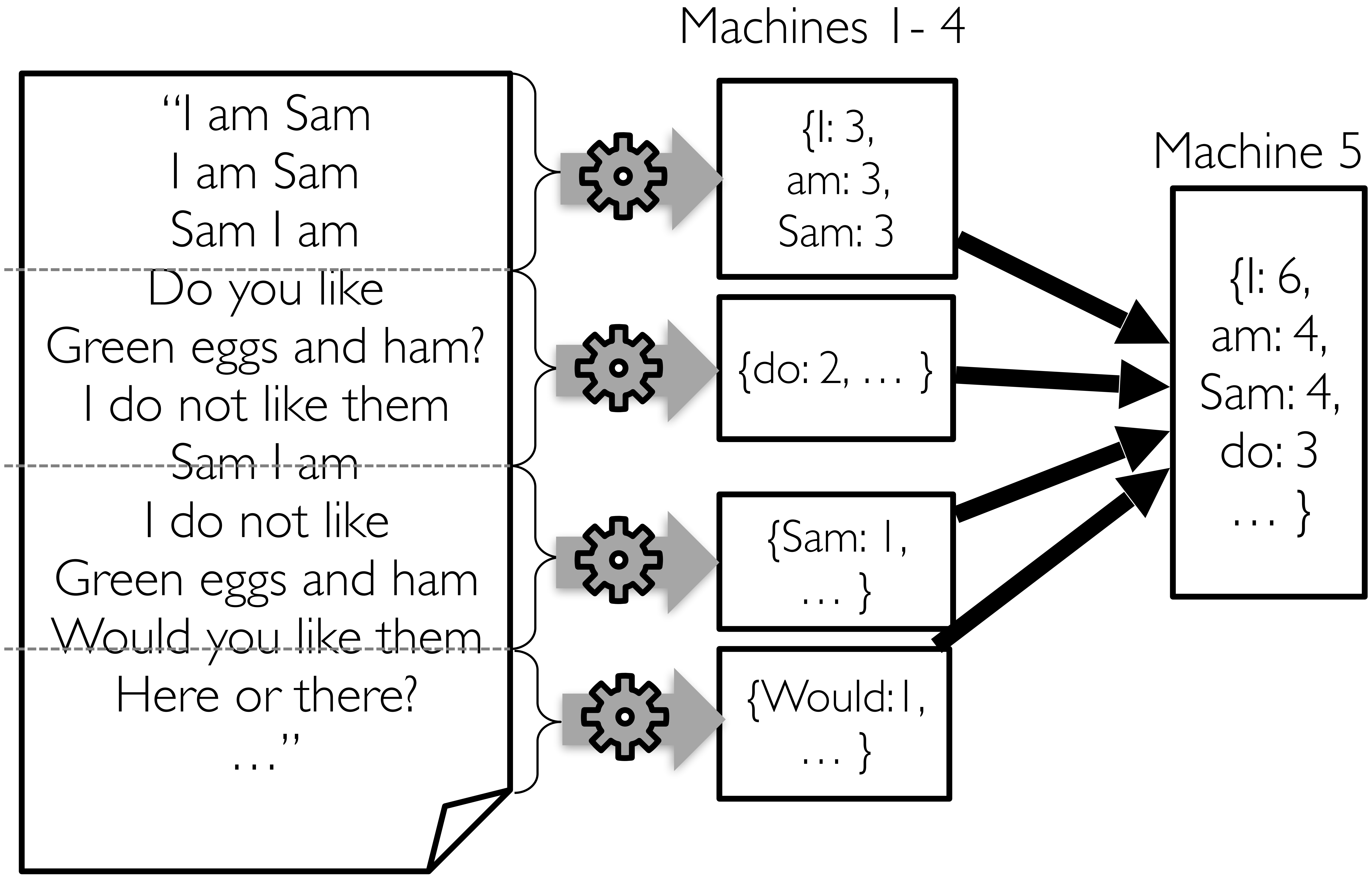
{Sam: 1,
… }

{Would:1,
… }

Machine 5

{I: 6,
am: 4,
Sam: 4,
do: 3
… }

What if the Document is Really Big?



“I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…”

Machines 1- 4

{I: 3,
am: 3,
Sam: 3

{do: 2, … }

{Sam: 1,
… }

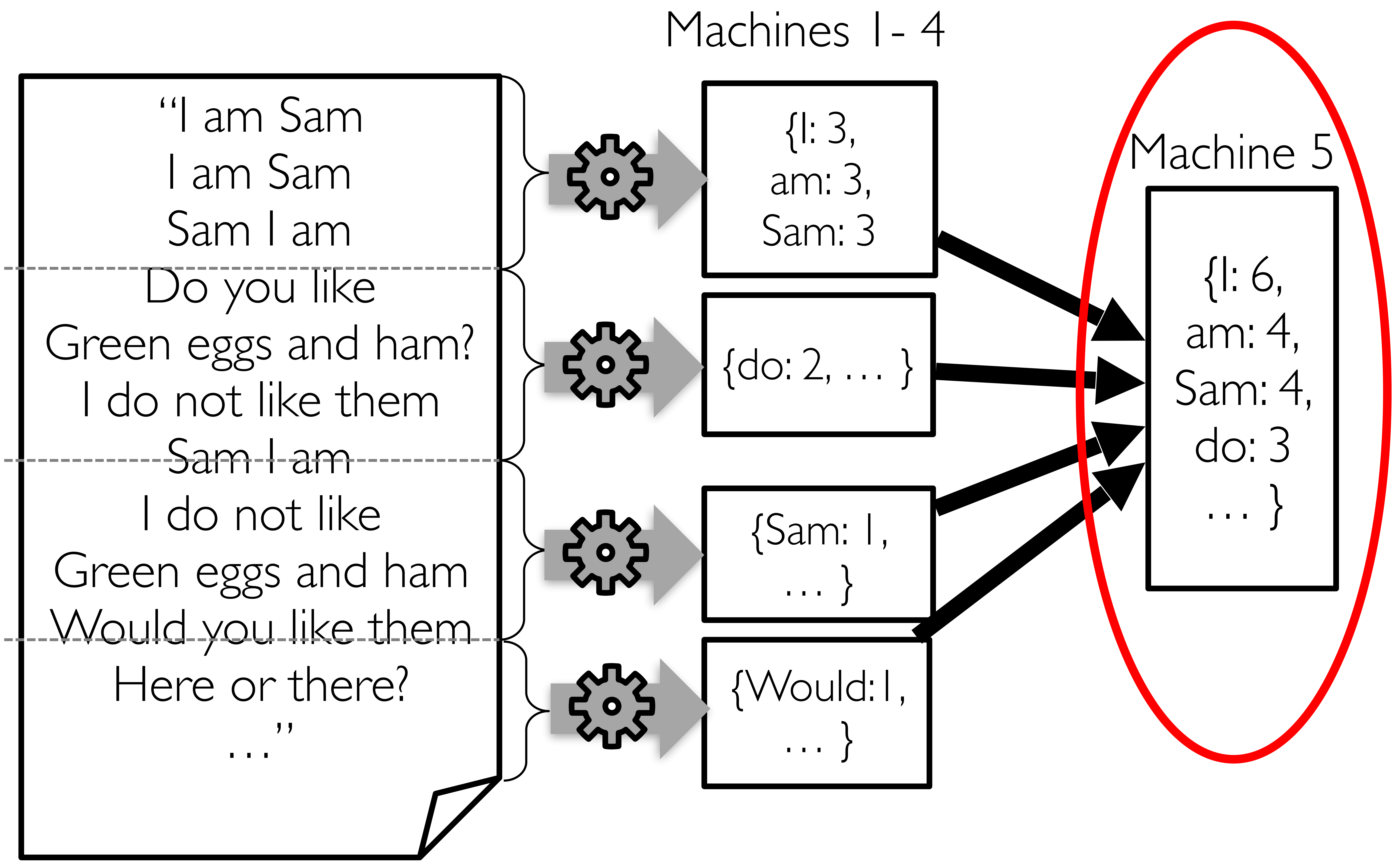
{Would:1,
… }

Machine 5

{I: 6,
am: 4,
Sam: 4,
do: 3
… }

What’s the problem with this approach?

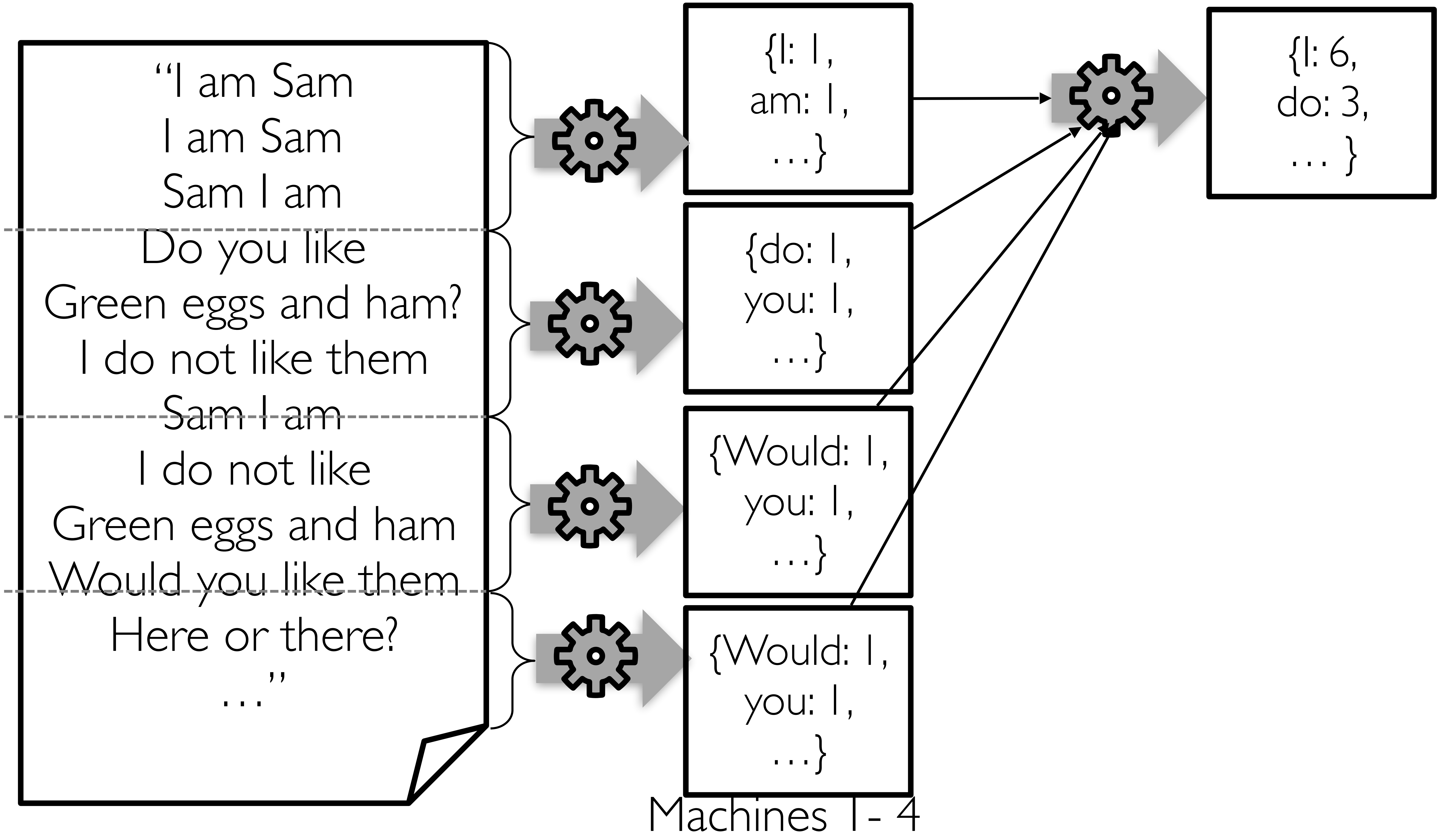
What if the Document is Really Big?



Machines 1- 4
"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"
{I: 3,
am: 3,
Sam: 3
{do: 2, … }
{Sam: 1,
… }
{Would:1,
… }
Machine 5
{I: 6,
am: 4,
Sam: 4,
do: 3
… }

Results have to fit on one machine

What if the Document is Really Big?



"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"

{I: 1,
am: 1,
…}

{do: 1,
you: 1,
…}

{Would: 1,
you: 1,
…}

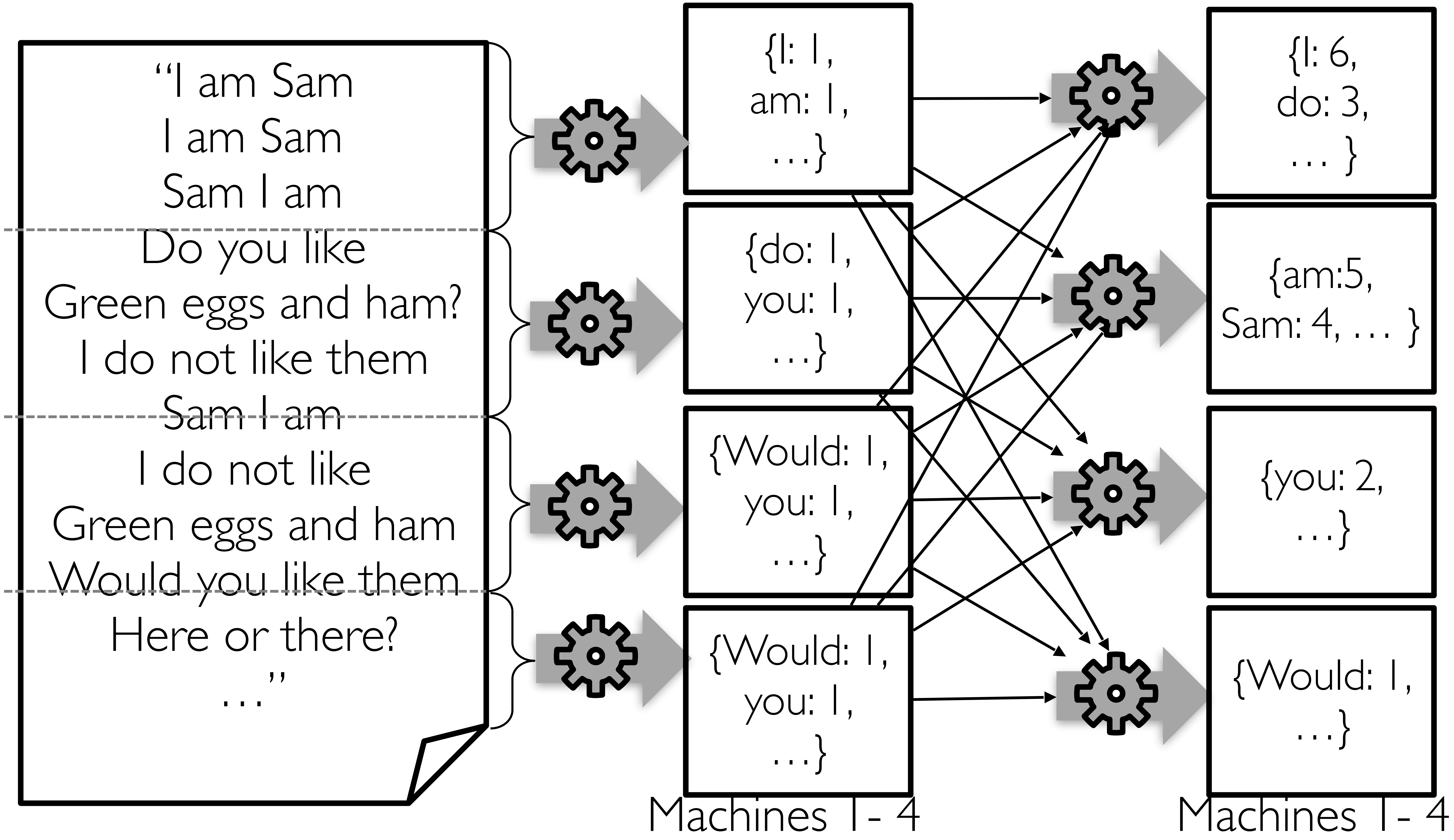
{Would: 1,
you: 1,
…}

Machines 1- 4

{I: 6,
do: 3,
… }

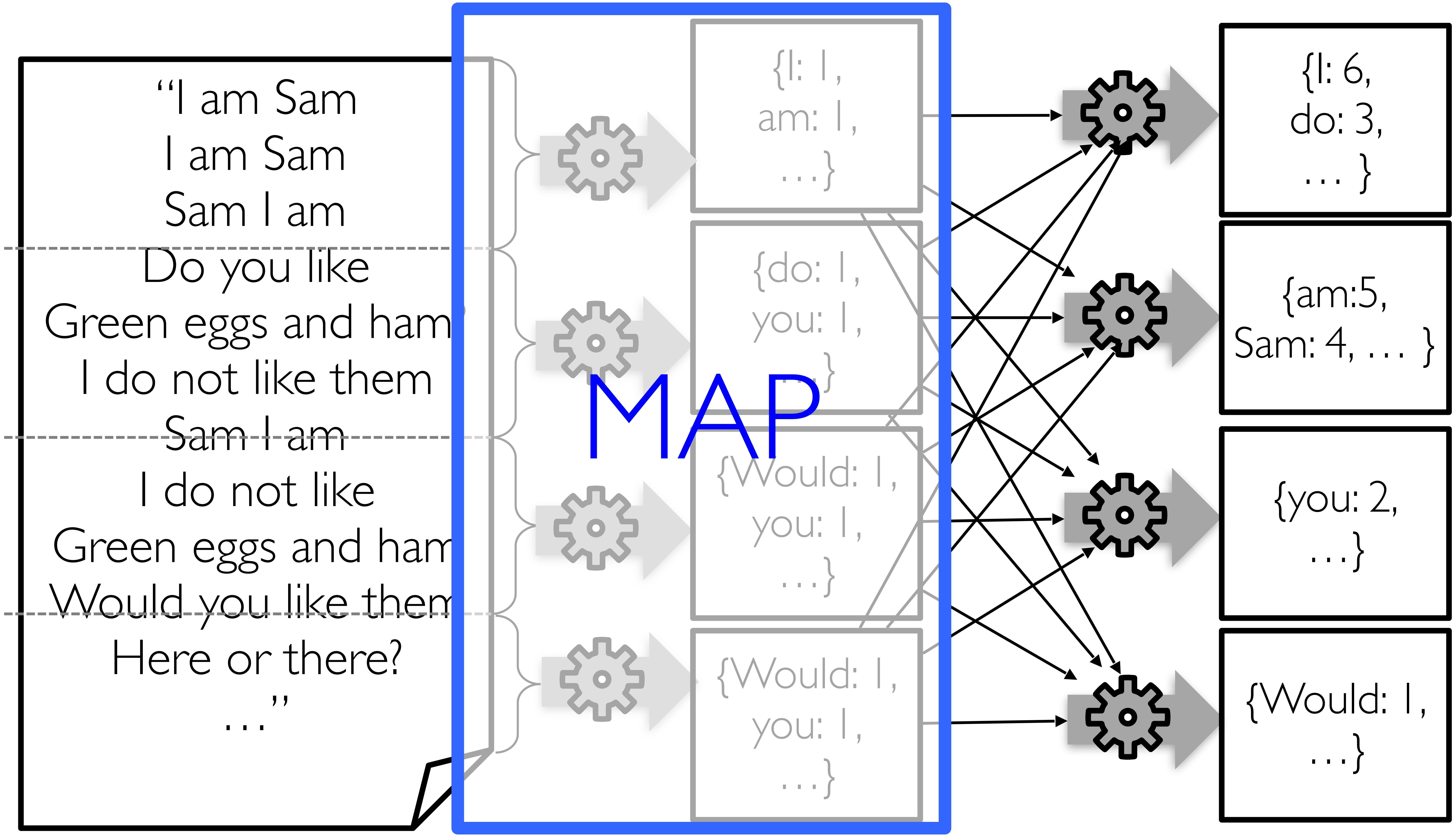
Use Divide and Conquer!!

What if the Document is Really Big?


"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"
{I: 1, am: 1, …}
{do: 1, you: 1, …}
{Would: 1, you: 1, …}
{Would: 1, you: 1, …}
Machines 1- 4
{I: 6, do: 3, … }
{am:5, Sam: 4, … }
{you: 2, …}
{Would: 1, …}
Machines 1- 4

Use Divide and Conquer!!

What if the Document is Really Big?



"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"

MAP

{I: 1,
am: 1,
…}

{do: 1,
you: 1,
…}

{Would: 1,
you: 1,
…}

{Would: 1,
you: 1,
…}

{I: 6,
do: 3,
… }

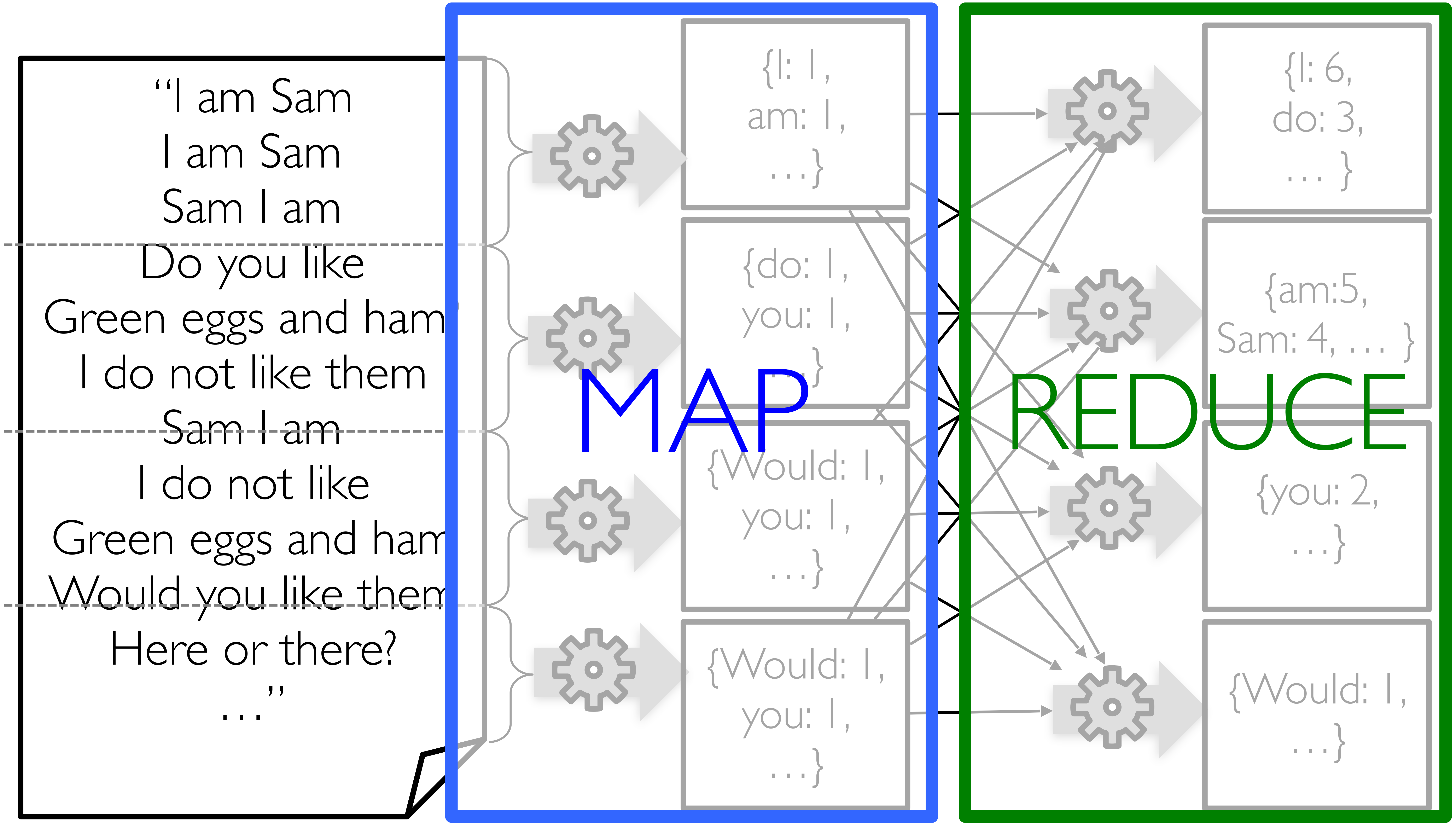
{am:5,
Sam: 4, … }

{you: 2,
…}

{Would: 1,
…}

Use Divide and Conquer!!

What if the Document is Really Big?



"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"

MAP

{I: 1,
am: 1,
…}

{do: 1,
you: 1,
…}

{Would: 1,
you: 1,
…}

{Would: 1,
you: 1,
…}

REDUCE

{I: 6,
do: 3,
… }

{am:5,
Sam: 4, … }

{you: 2,
…}

{Would: 1,
…}

Google
Map Reduce 2004

Apache Hadoop

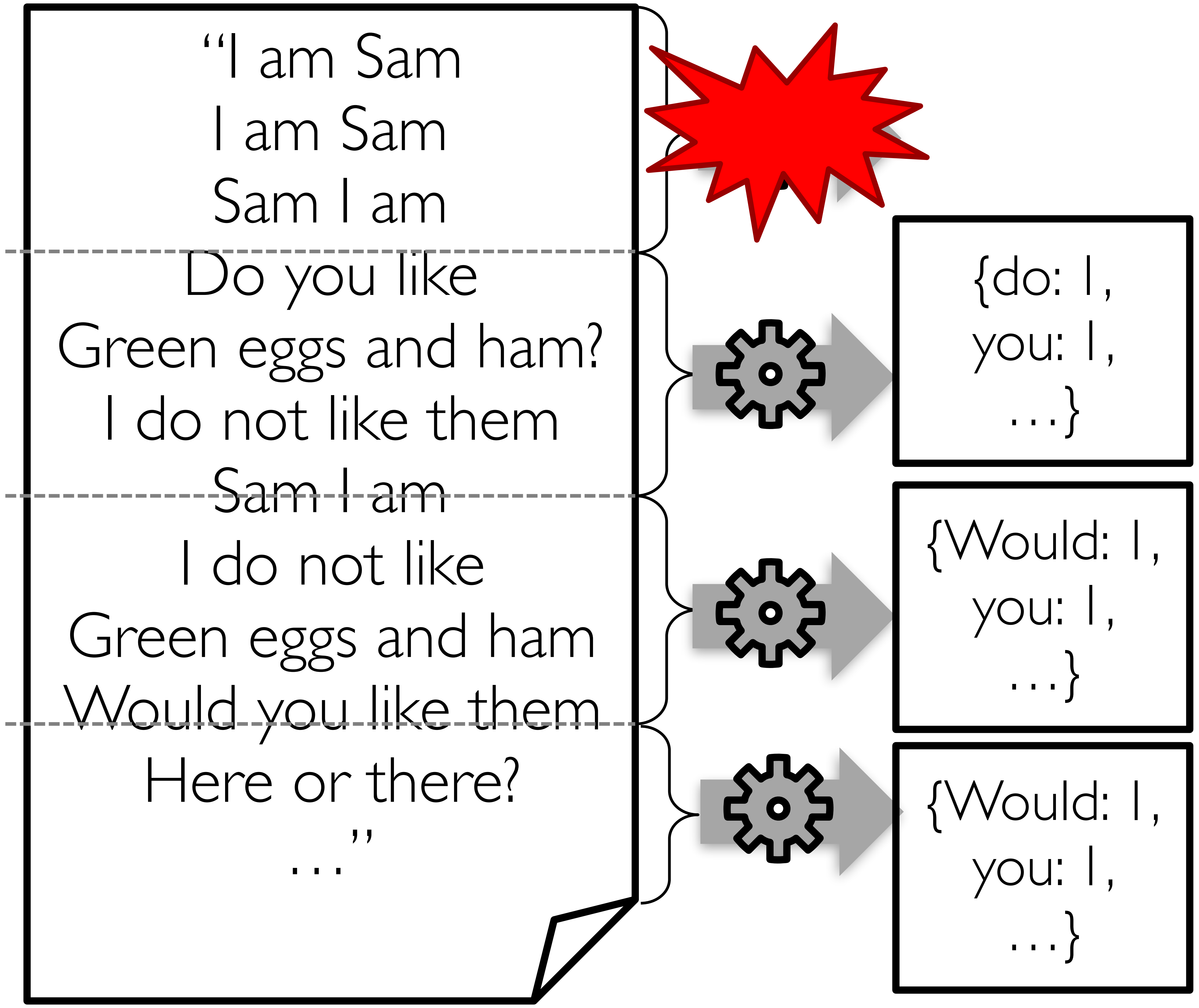
http://research.google.com/archive/mapreduce.html

What's Hard About Cluster Computing?

How to deal with failures?

- » 1 server fails every 3 years ⇒ with 10,000 nodes see 10 faults/day
- » Even worse: stragglers (not failed, but slow nodes)

How Do We Deal with Failures?



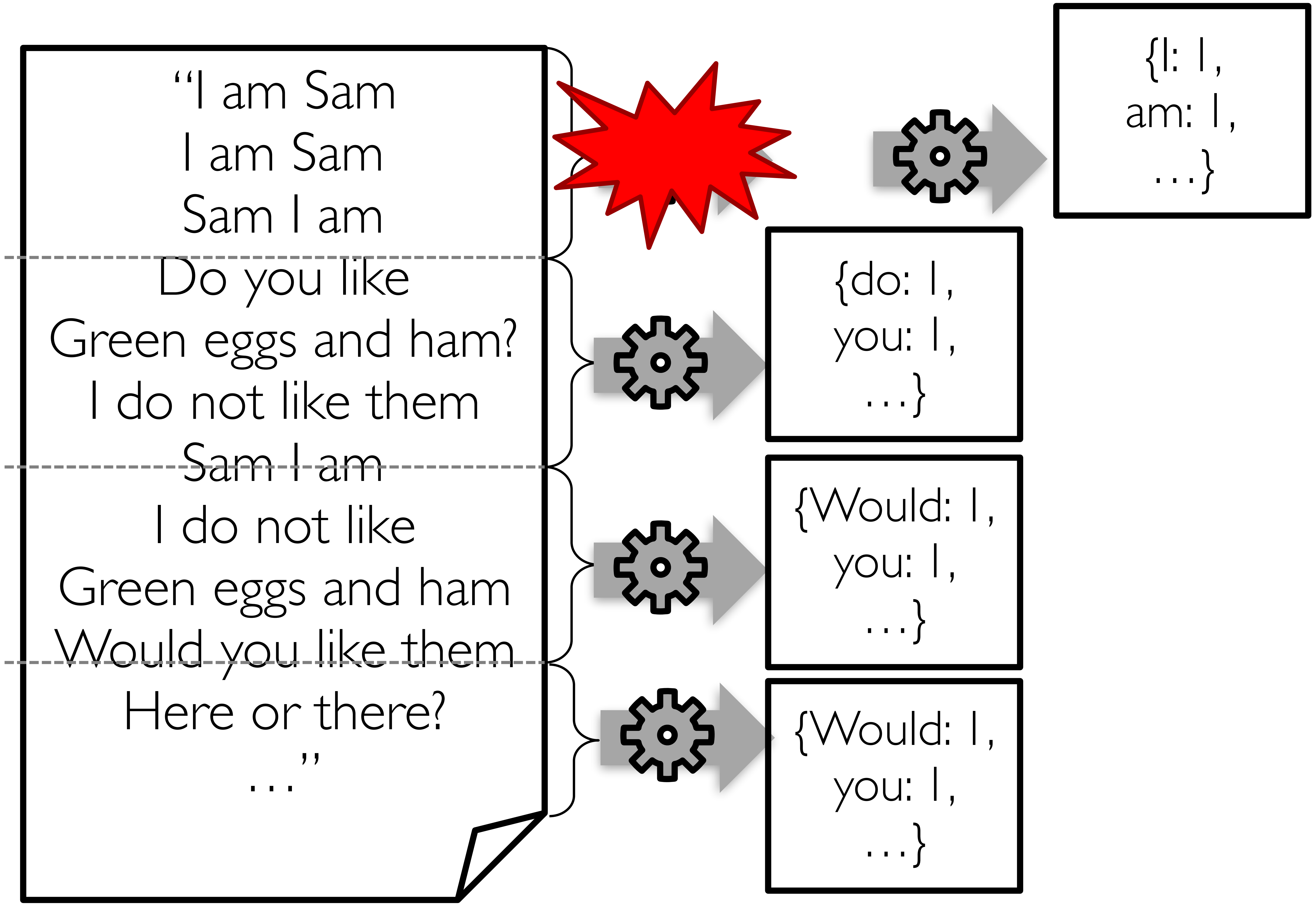
"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"

{do: I,
you: I,
…}

{Would: I,
you: I,
…}

{Would: I,
you: I,
…}

How Do We Deal with Machine Failures?



"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"

{I: 1,
am: 1,
…}

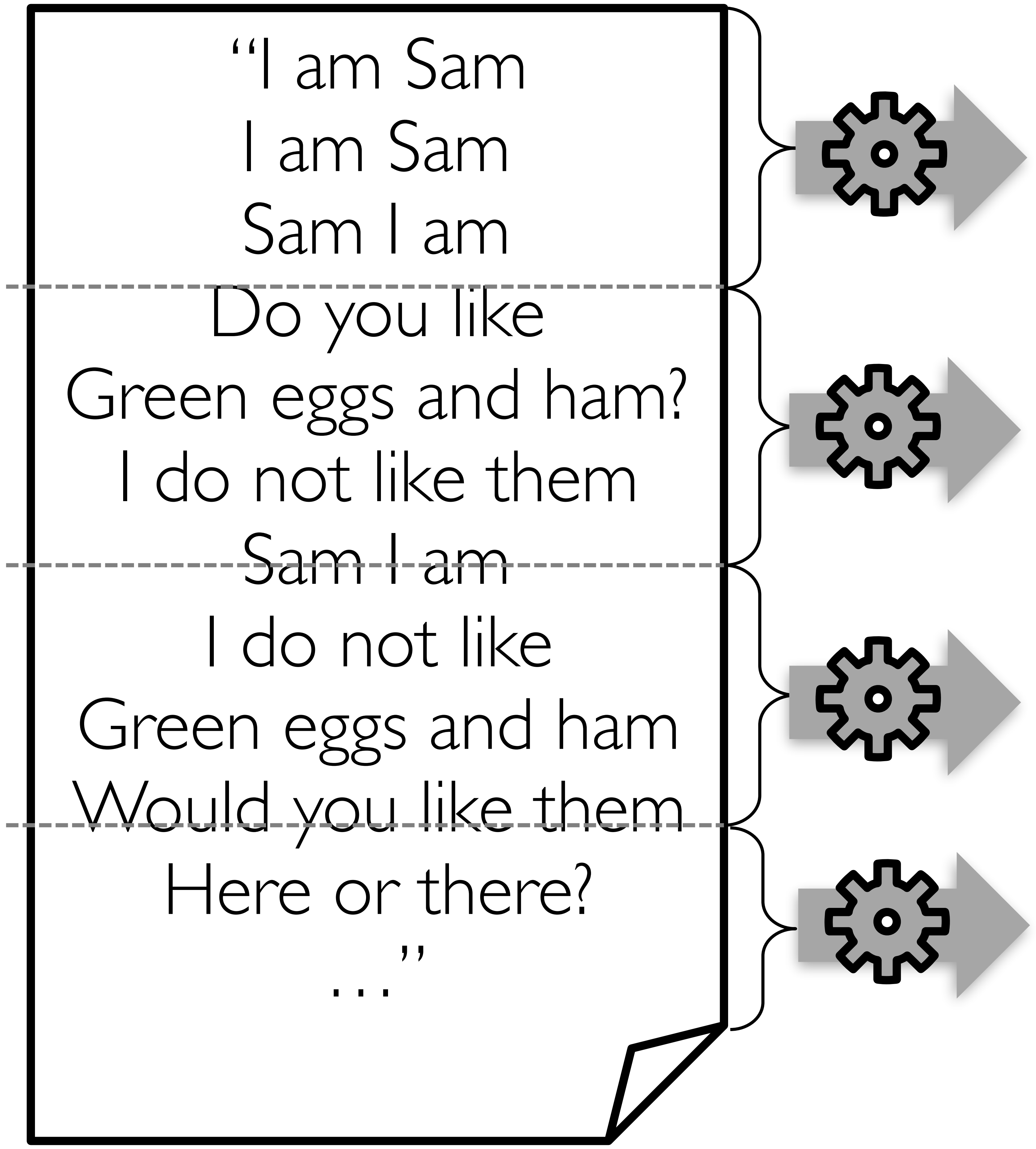
{do: 1,
you: 1,
…}

{Would: 1,
you: 1,
…}

{Would: 1,
you: 1,
…}

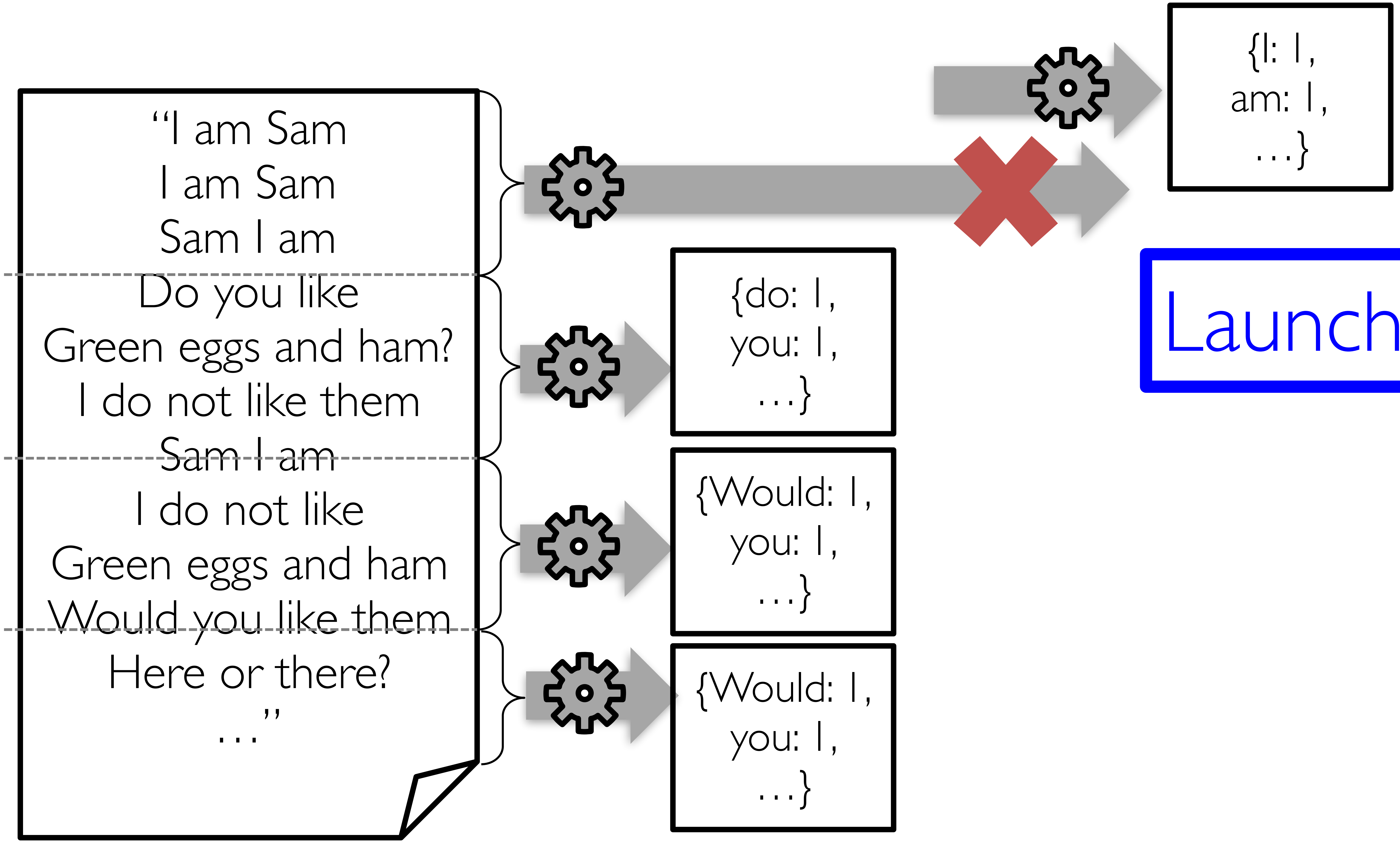
Launch another task!

How Do We Deal with Slow Tasks?



"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"

How Do We Deal with Slow Tasks?



"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?
I do not like them
Sam I am
I do not like
Green eggs and ham
Would you like them
Here or there?
…"

{I: 1,
am: 1,
…}

{do: 1,
you: 1,
…}

{Would: 1,
you: 1,
…}

{Would: 1,
you: 1,
…}

Launch another task!

What’s Hard About Cluster Computing?

How to deal with failures?

- » 1 server fails every 3 years ⇒ with 10,000 nodes see 10 faults/day
- » Even worse: stragglers (not failed, but slow nodes)

How to divide work across machines?

- » Must consider network, data locality
- » Moving data may be very expensive

Outline

1. Big Data: Challenges and Opportunities
2. MapReduce / Hadoop
3. Datacenter Communication Hierarchy
4. Spark and ML
5. HW1 preview

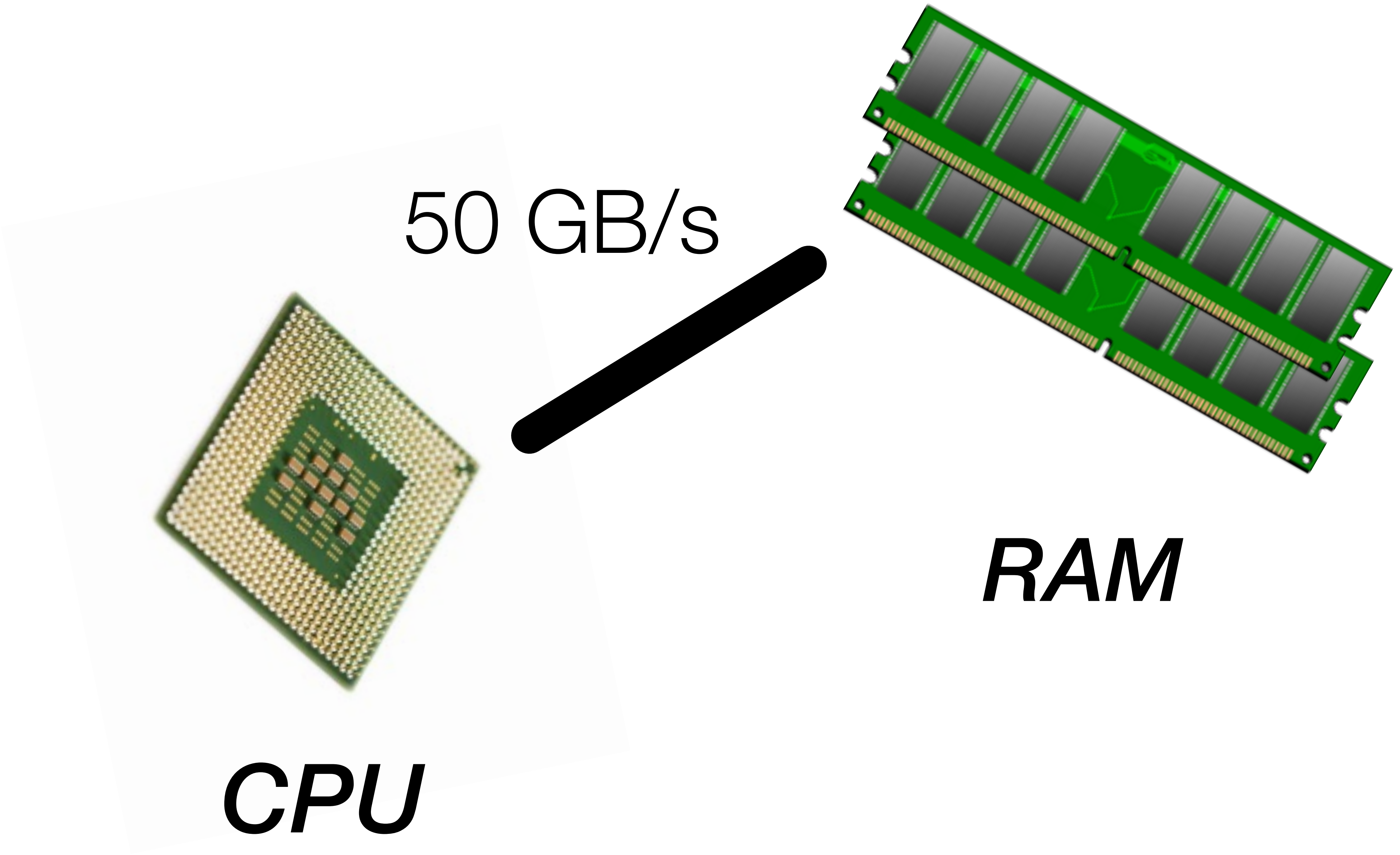
Communication Hierarchy

CPU

2 billion cycles/sec per core

Clock speeds not changing, but number of cores growing with Moore's Law

Communication Hierarchy

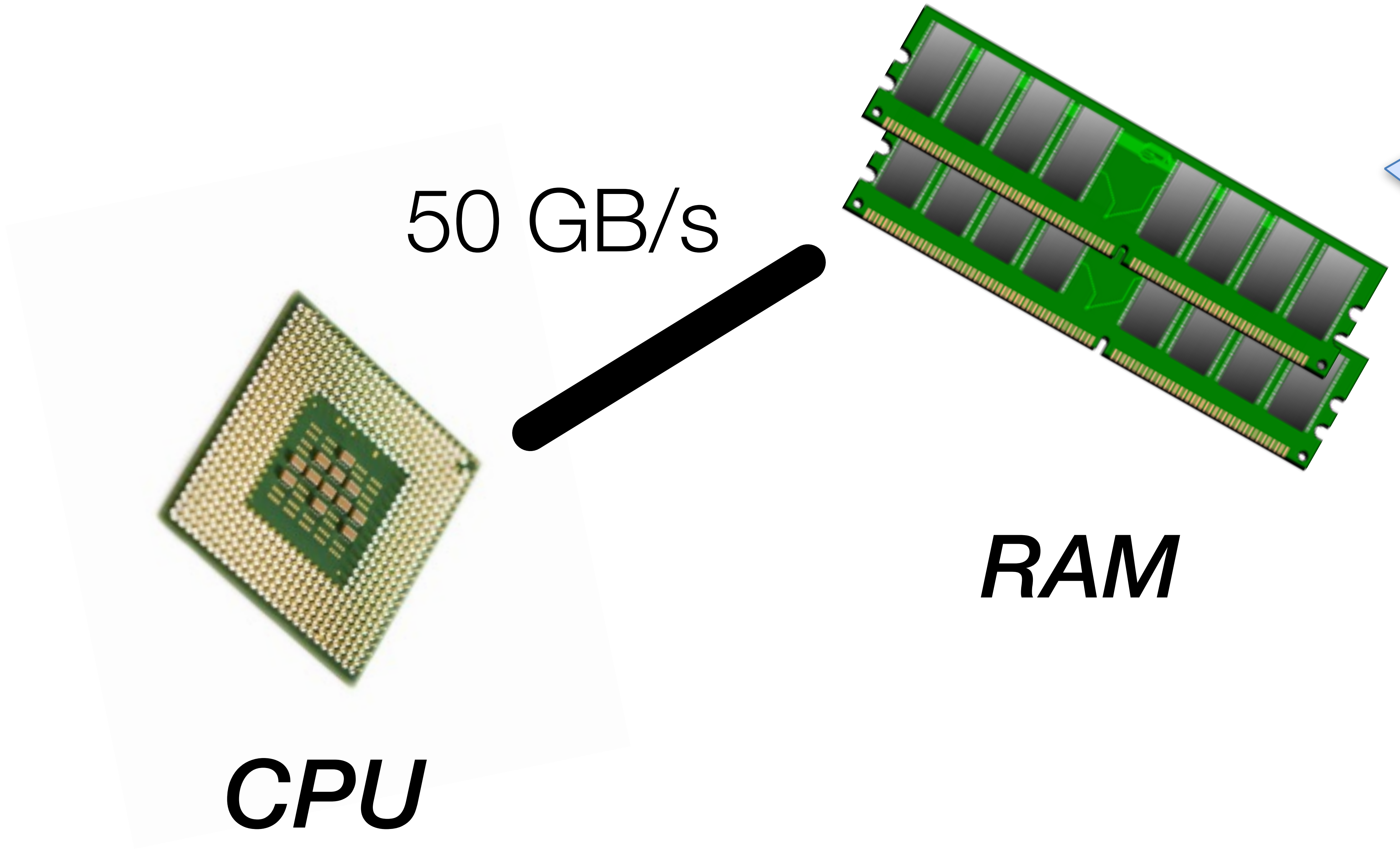


50 GB/s

RAM

CPU

Communication Hierarchy



50 GB/s

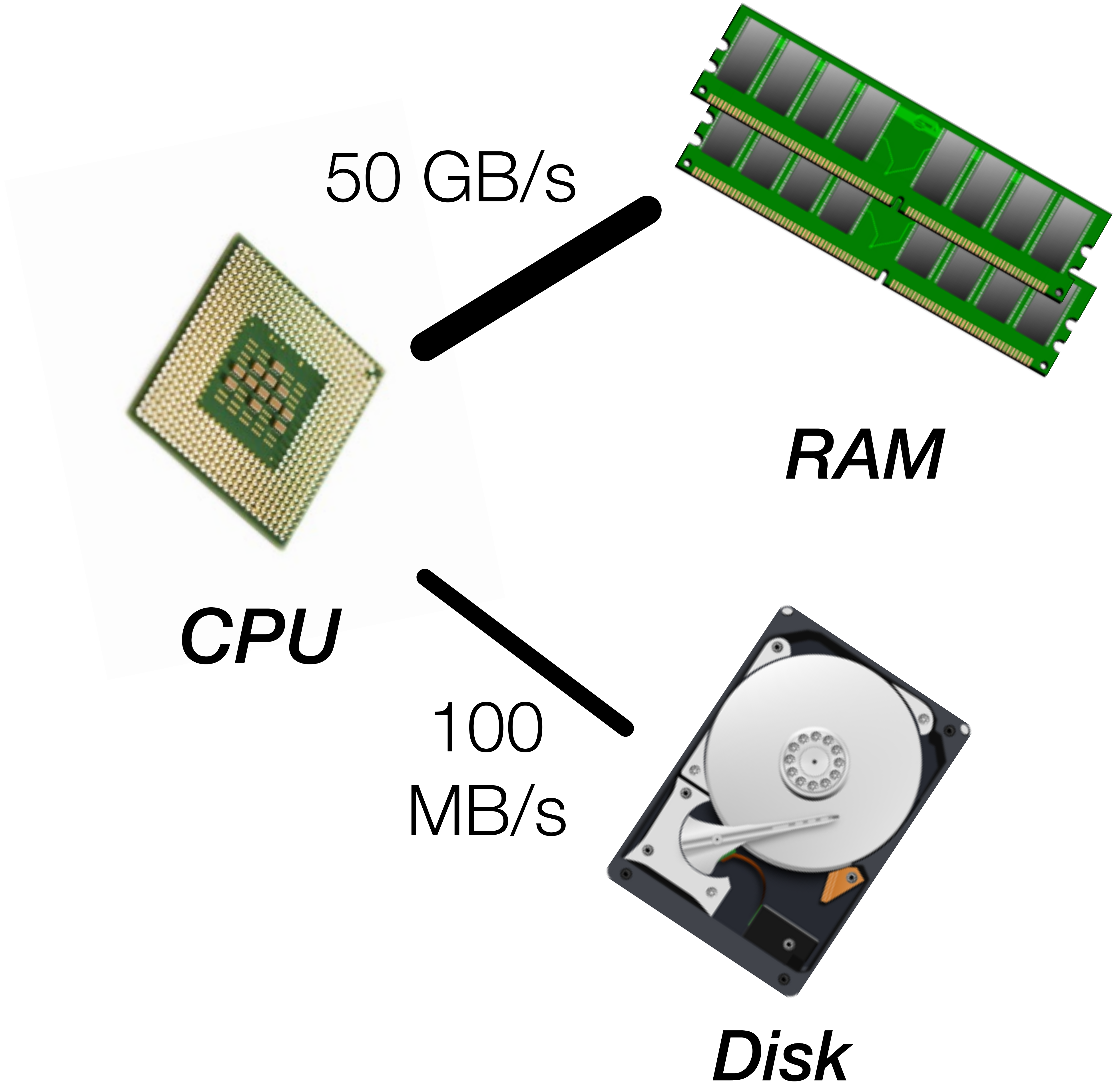
RAM

CPU

10-100 GB

Capacity growing with Moore’s Law

Communication Hierarchy



50 GB/s

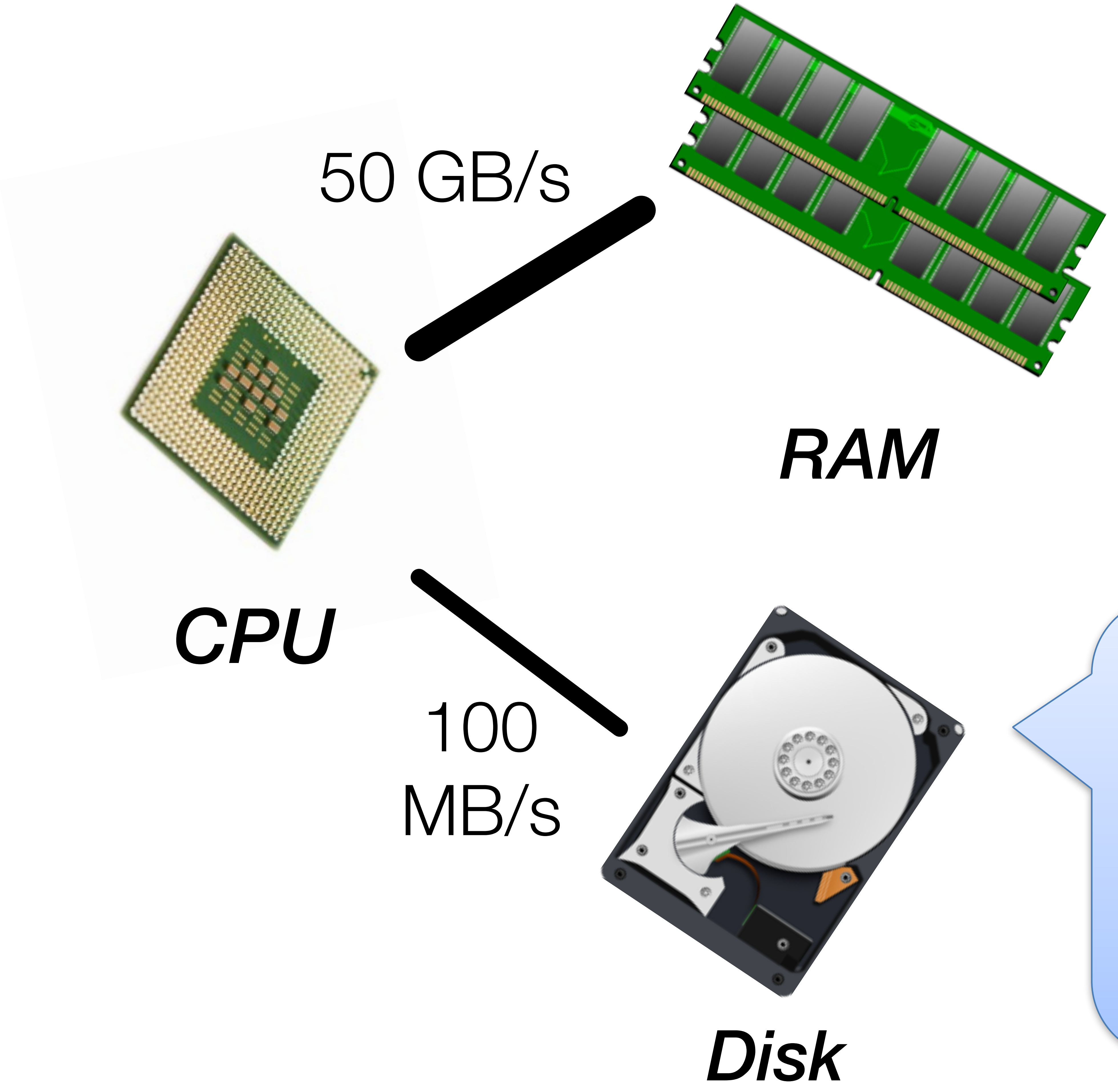
RAM

CPU

100 MB/s

Disk

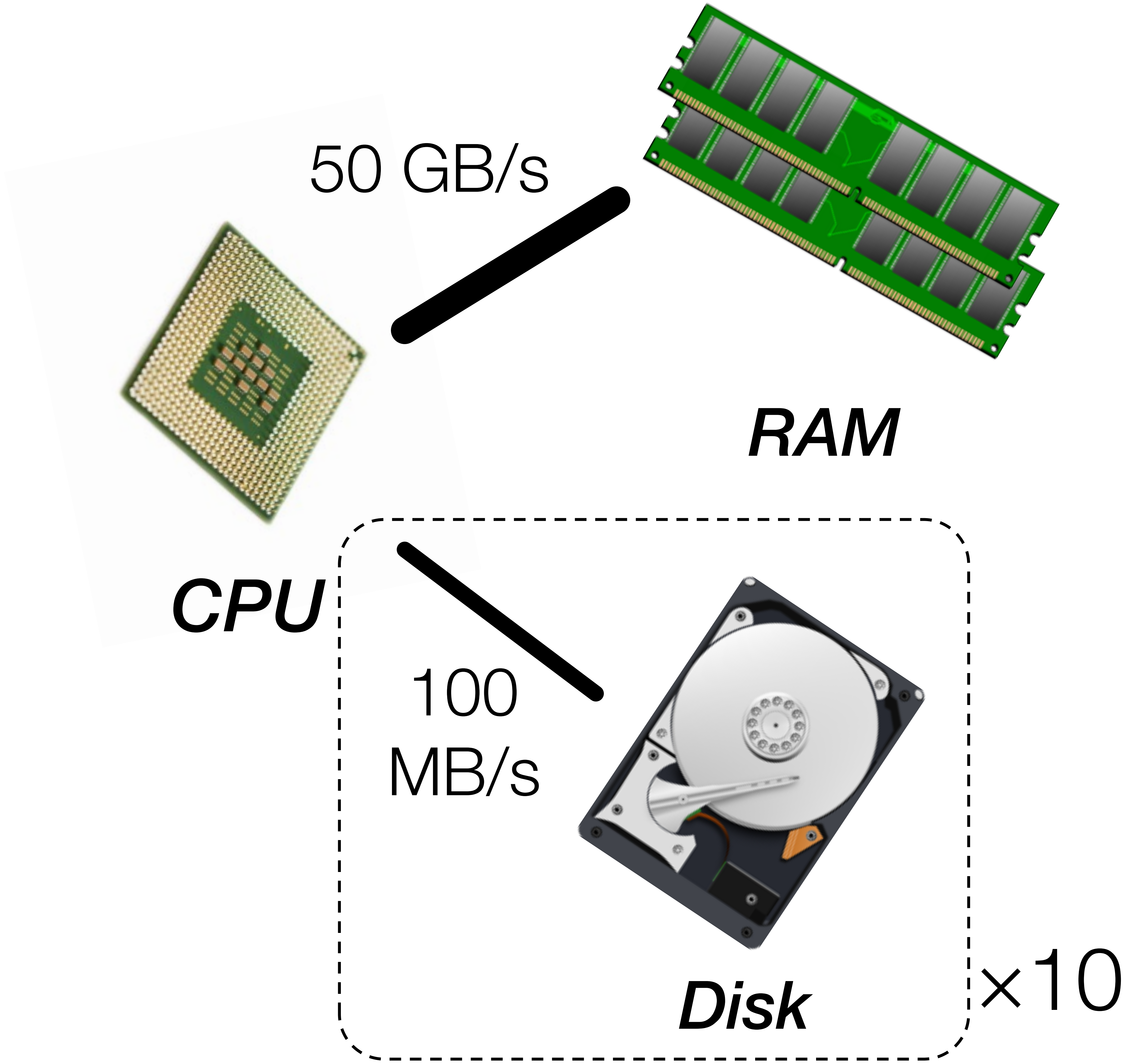
Communication Hierarchy


50 GB/s
RAM
CPU
100
MB/s
Disk

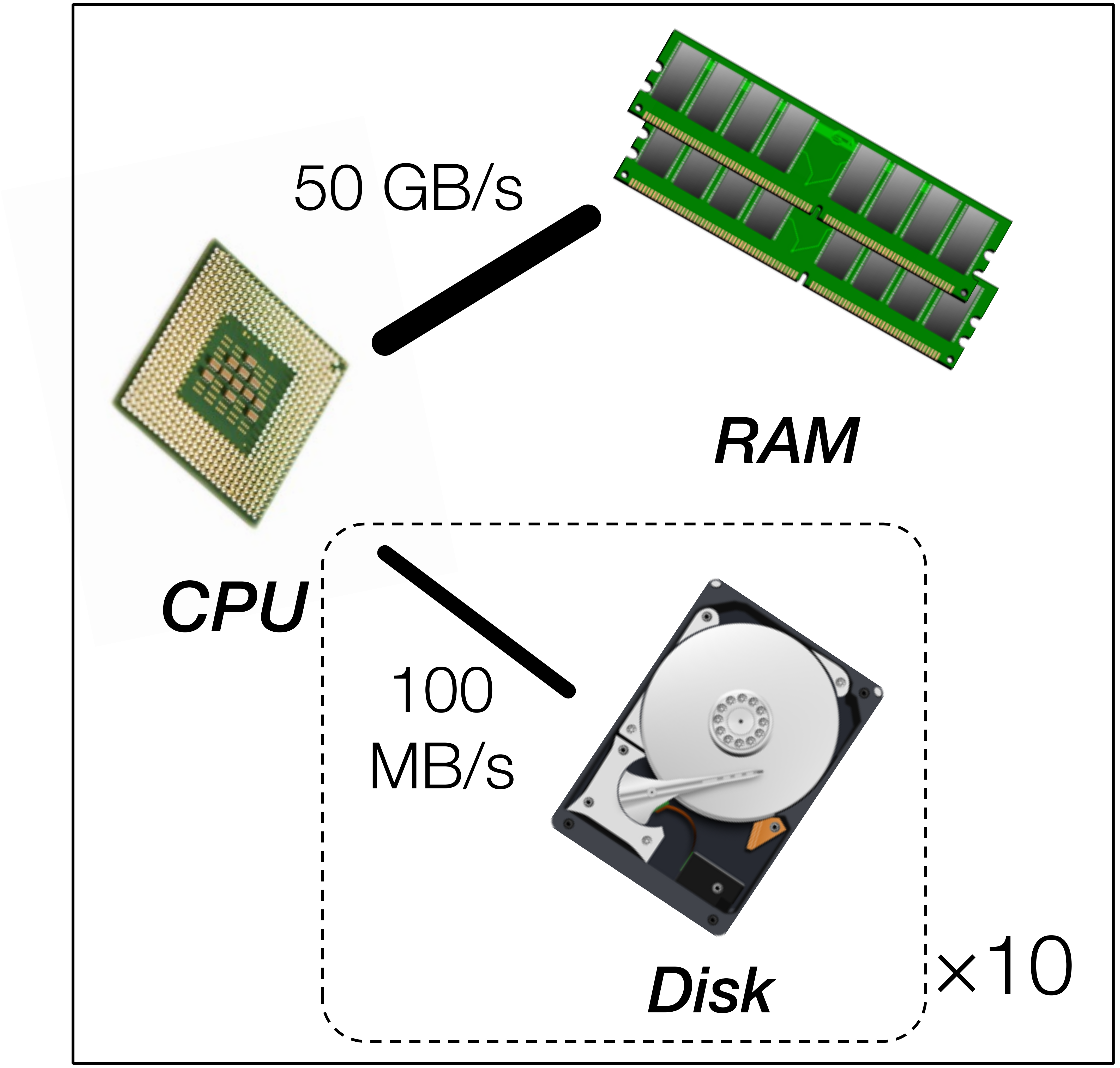
1-2 TB

Capacity growing exponentially, but not speed

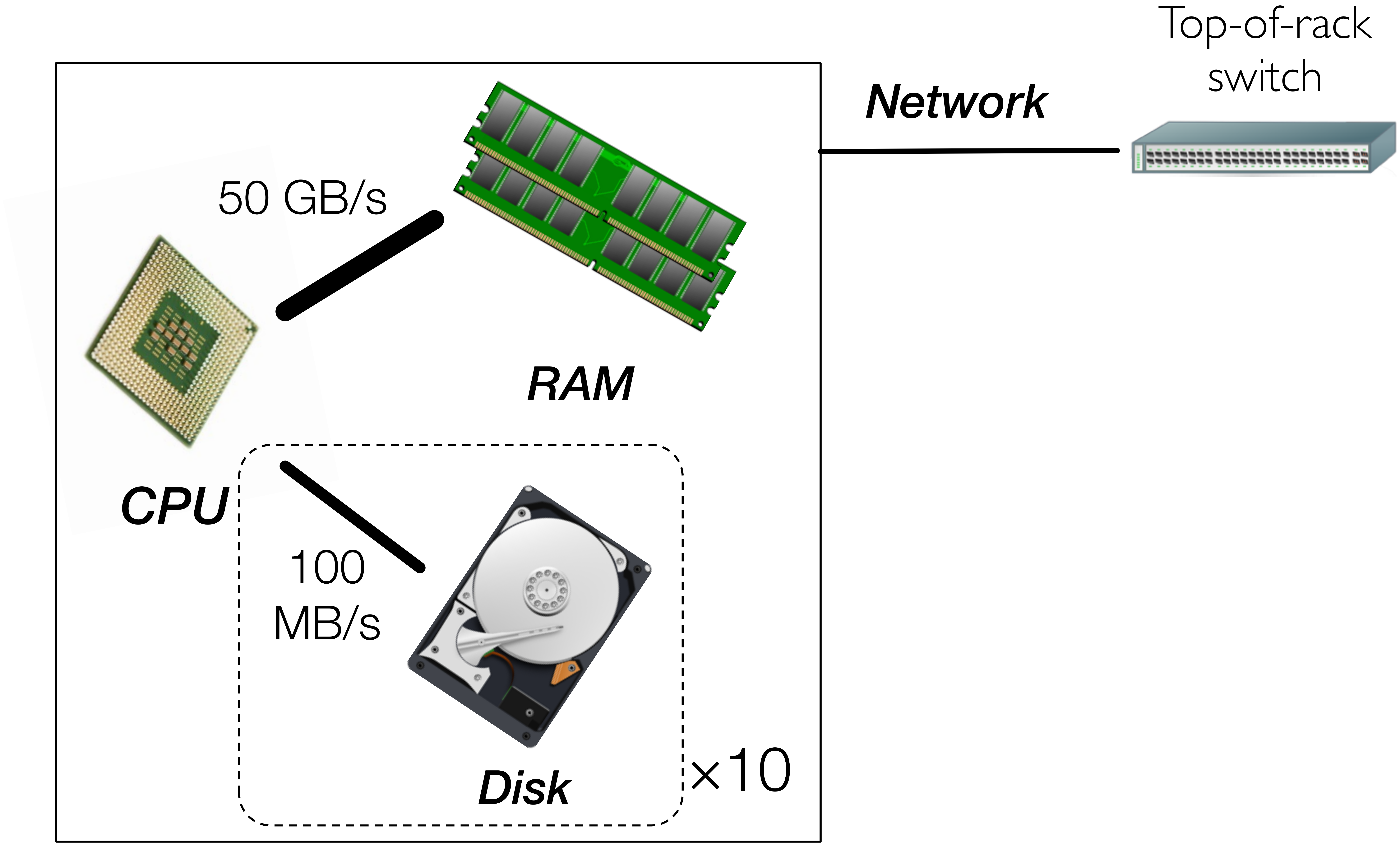
Communication Hierarchy


50 GB/s
RAM
CPU
100
MB/s
Disk
×10

Communication Hierarchy


50 GB/s
RAM
CPU
100
MB/s
Disk
×10

Communication Hierarchy



Top-of-rack switch

Network

50 GB/s

RAM

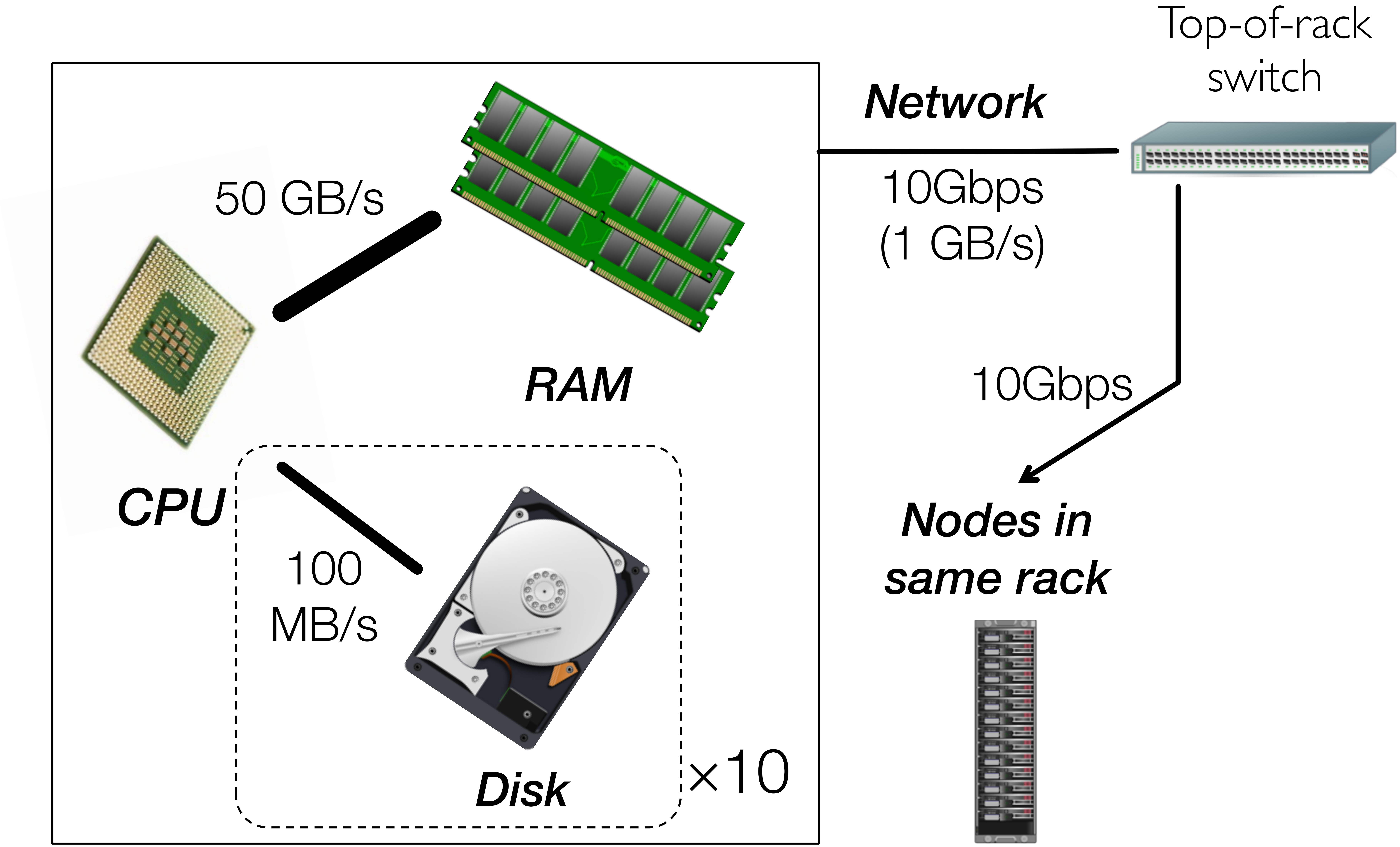
CPU

100 MB/s

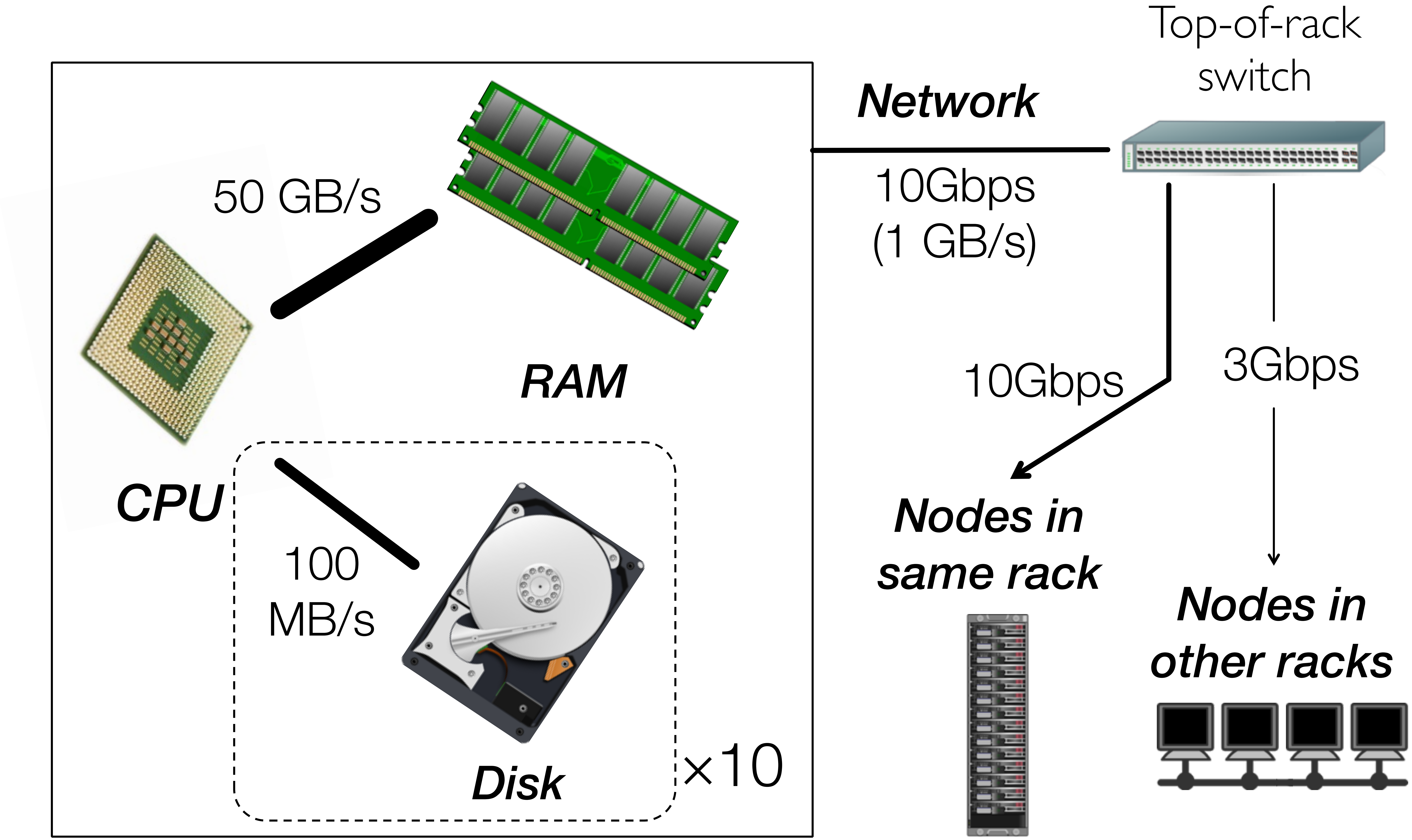
Disk

×10

Communication Hierarchy


Top-of-rack
switch
Network
10Gbps
(1 GB/s)
50 GB/s
RAM
CPU
100
MB/s
Disk
×10
10Gbps
Nodes in
same rack

Communication Hierarchy


50 GB/s
RAM
CPU
100 MB/s
Disk
×10
Network
10Gbps (1 GB/s)
Top-of-rack switch
10Gbps
3Gbps
Nodes in same rack
Nodes in other racks

Key Takeaway: Access rates fall sharply with distance!

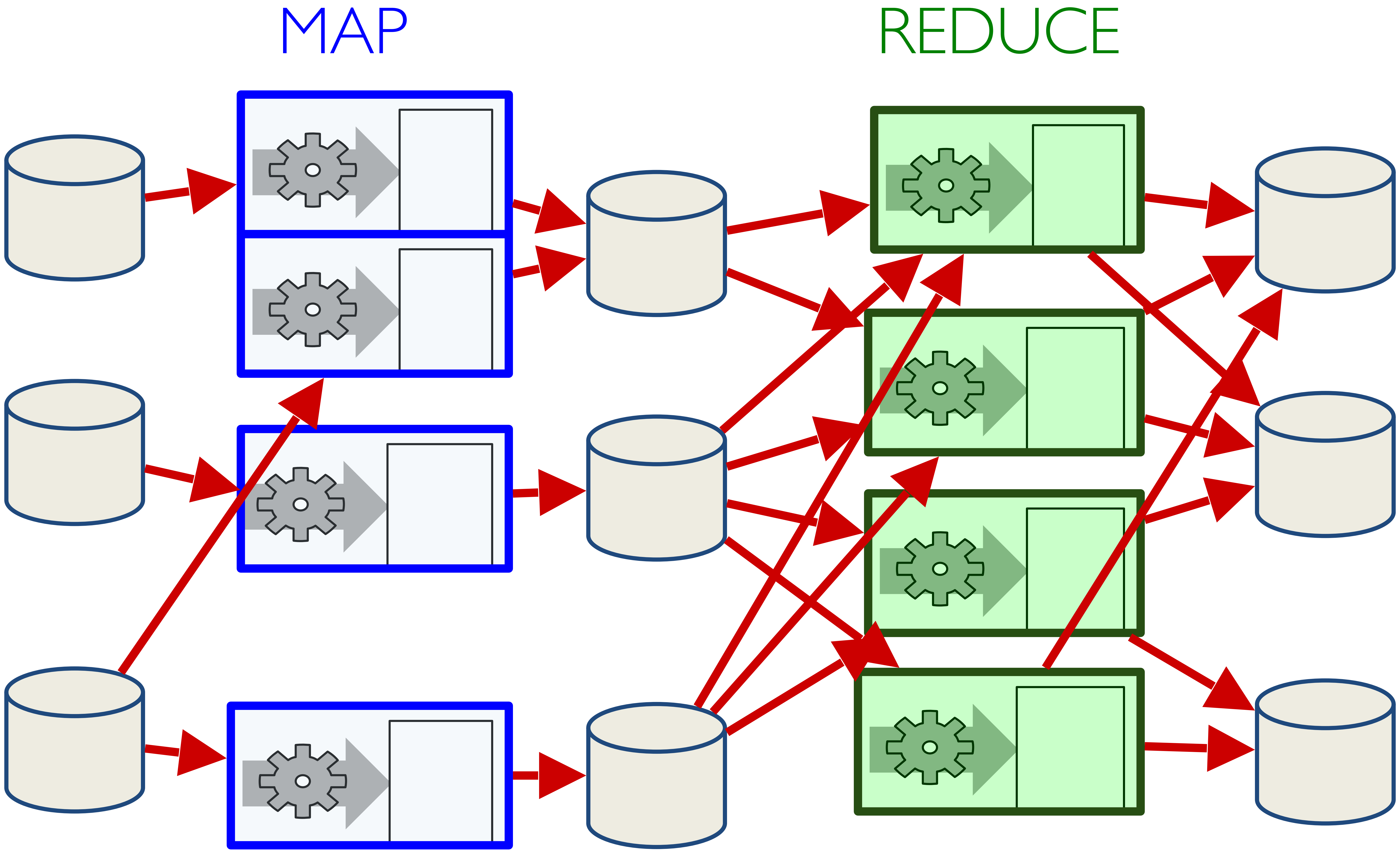
50x gap between Memory and Disk

Network is on average slower than Disk

50 GB/s
1 GB/s
1 GB/s
0.3 GB/s
CPU
RAM
Local disks
Rack
Different Racks

Implication for Hadoop/MapReduce?

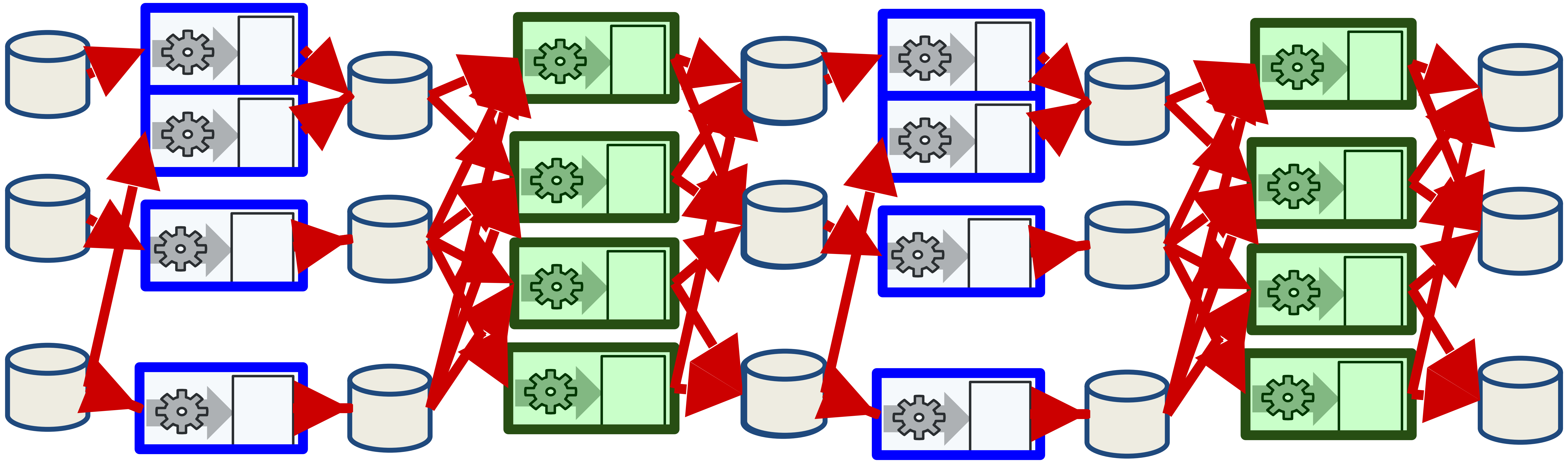
Map Reduce: Distributed Execution



Each stage passes through the hard drives

Map Reduce: Iterative Jobs

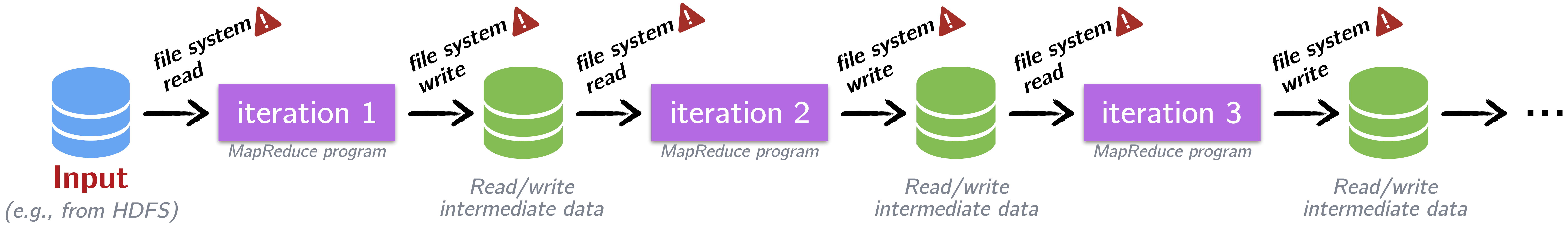
- Iterative jobs involve a lot of disk I/O for each repetition



Disk I/O is very slow!

Why do we care about iterative workloads in this class?

Iteration in Hadoop:



Input

(e.g., from HDFS)

file system read ⚠

iteration 1

MapReduce program

file system write ⚠

Read/write intermediate data

file system read ⚠

iteration 2

MapReduce program

file system write ⚠

Read/write intermediate data

file system read ⚠

iteration 3

MapReduce program

file system write ⚠

Read/write intermediate data

...

Outline

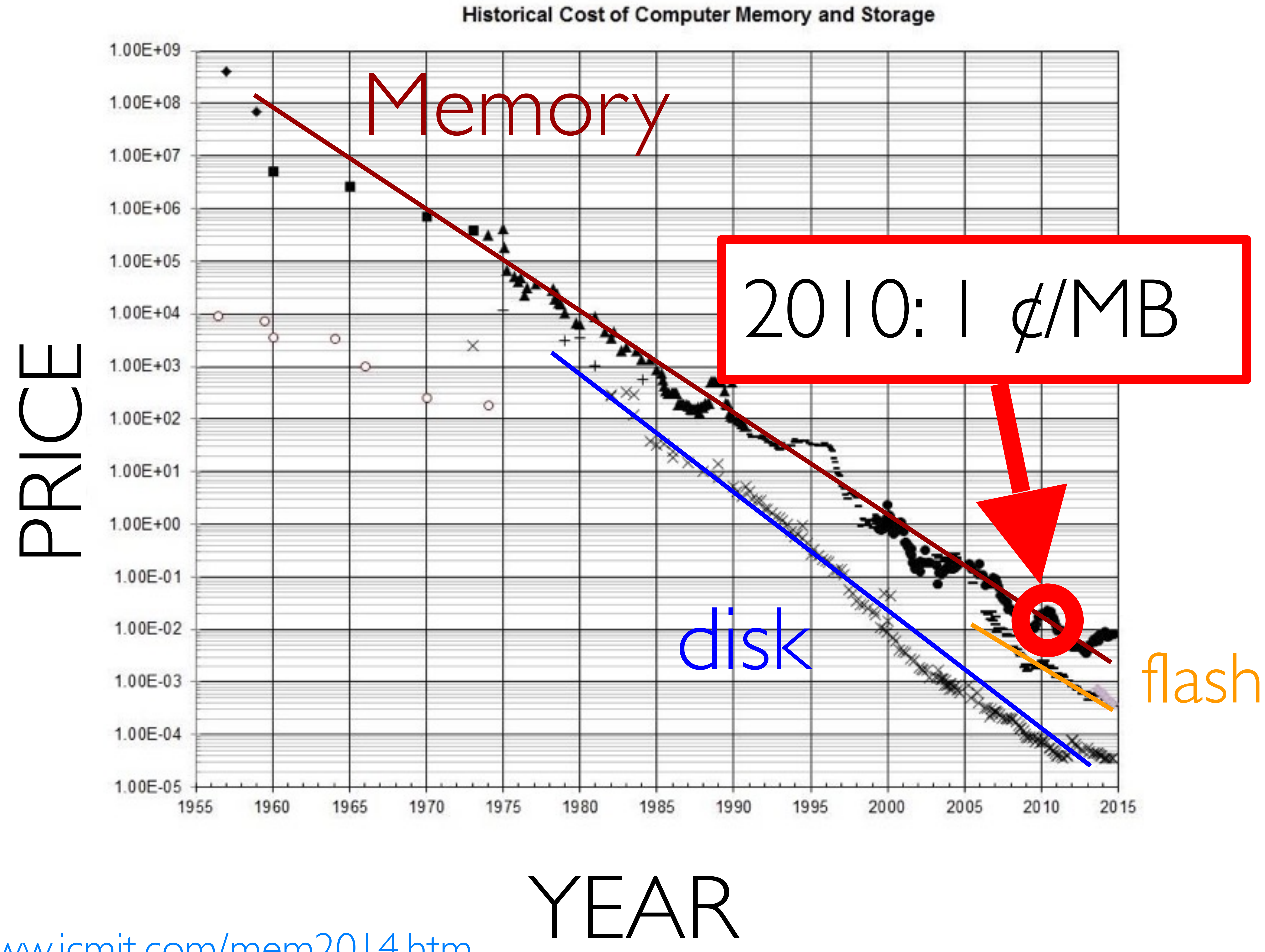
1. Big Data: Challenges and Opportunities
2. MapReduce / Hadoop
3. Datacenter Communication Hierarchy
4. Spark and ML
5. HW1 preview

Apache Spark Motivation

Using Map Reduce for complex jobs, interactive queries and online processing involves *lots of disk I/O*

Disk I/O is very slow

Tech Trend: Cost of Memory



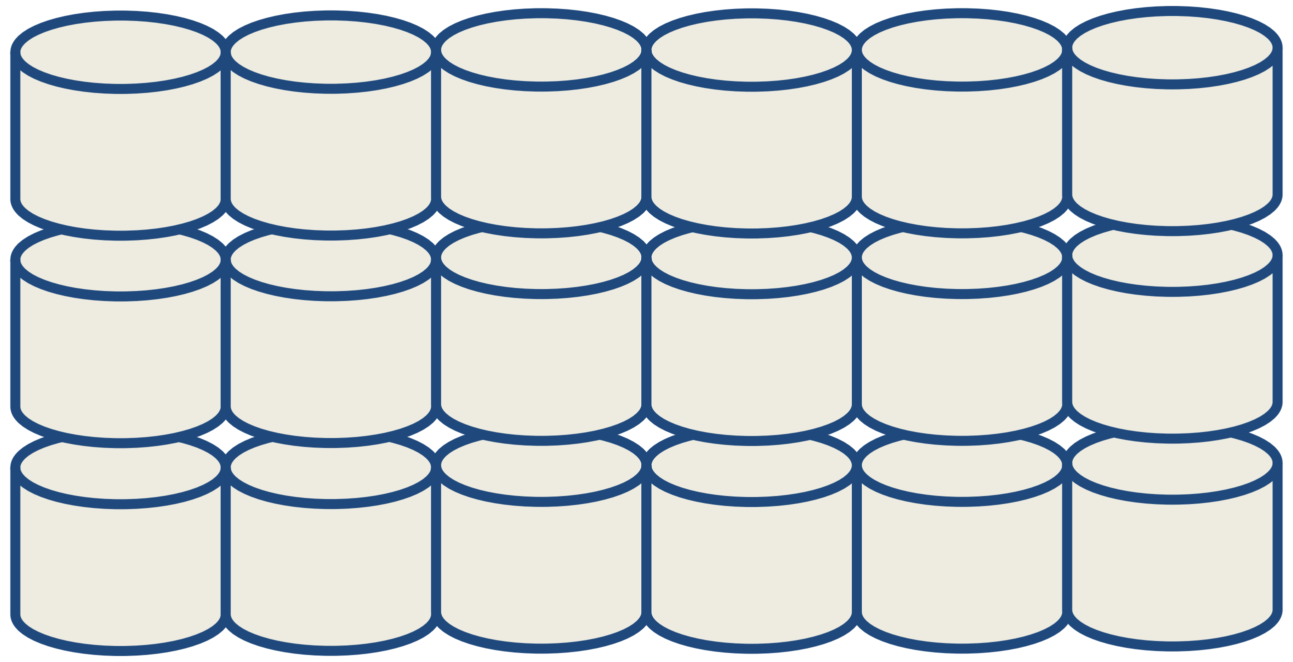
Historical Cost of Computer Memory and Storage

2010: 1 ¢/MB

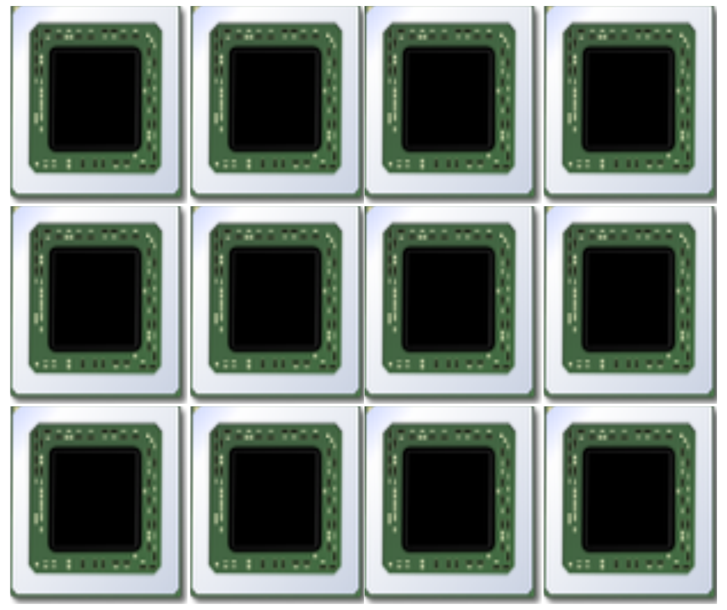
Lower cost means can put more memory in each server

http://www.jcmit.com/mem2014.htm

Modern Hardware for Big Data



Lots of hard drives



… and CPUs

… and memory!

Opportunity

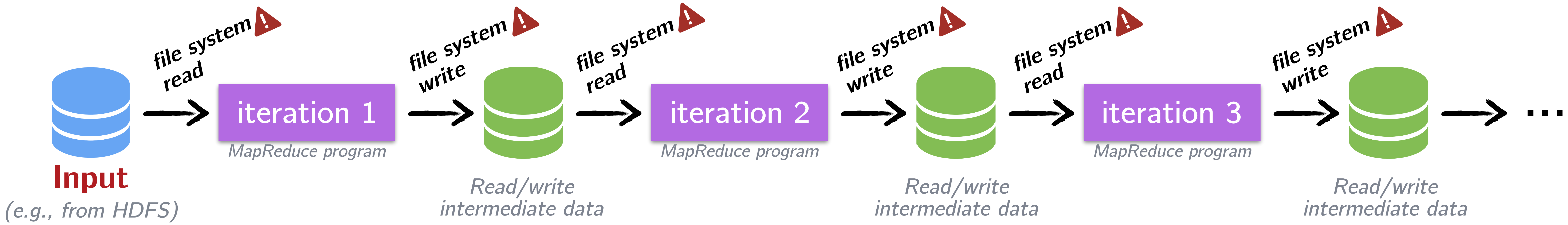
- Keep more data *in-memory*

- Create new distributed execution engine:

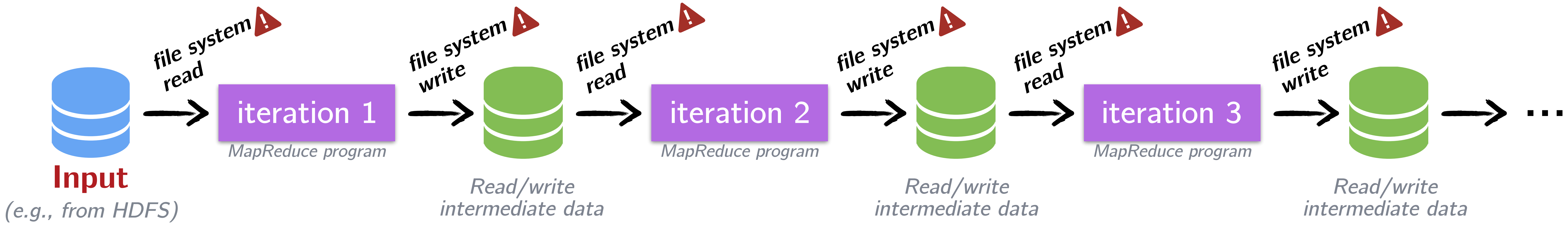
Spark

http://people.csail.mit.edu/matei/papers/2010/hotcloud_spark.pdf

Iteration in Hadoop:

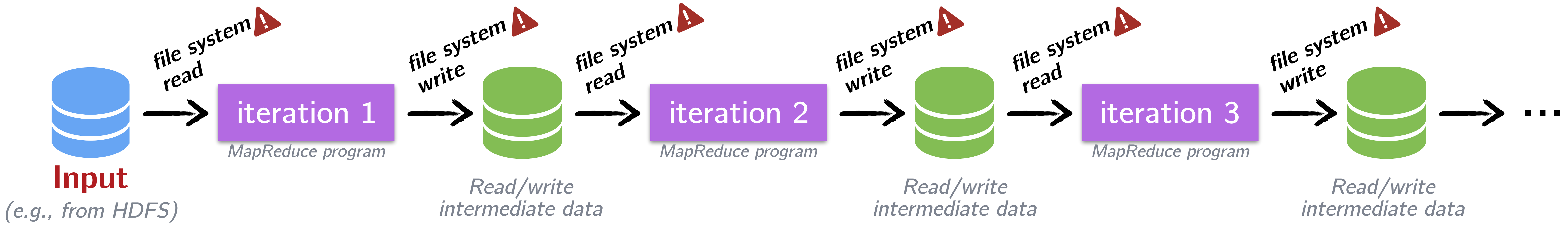

file system read
iteration 1
MapReduce program
file system write
file system read
iteration 2
MapReduce program
file system write
file system read
iteration 3
MapReduce program
file system write
…
Input
(e.g., from HDFS)
Read/write intermediate data
Read/write intermediate data
Read/write intermediate data

Iteration in Hadoop:


file system read
iteration 1
MapReduce program
file system write
file system read
iteration 2
MapReduce program
file system write
file system read
iteration 3
MapReduce program
file system write
...
Input
(e.g., from HDFS)
Read/write intermediate data
Read/write intermediate data
Read/write intermediate data

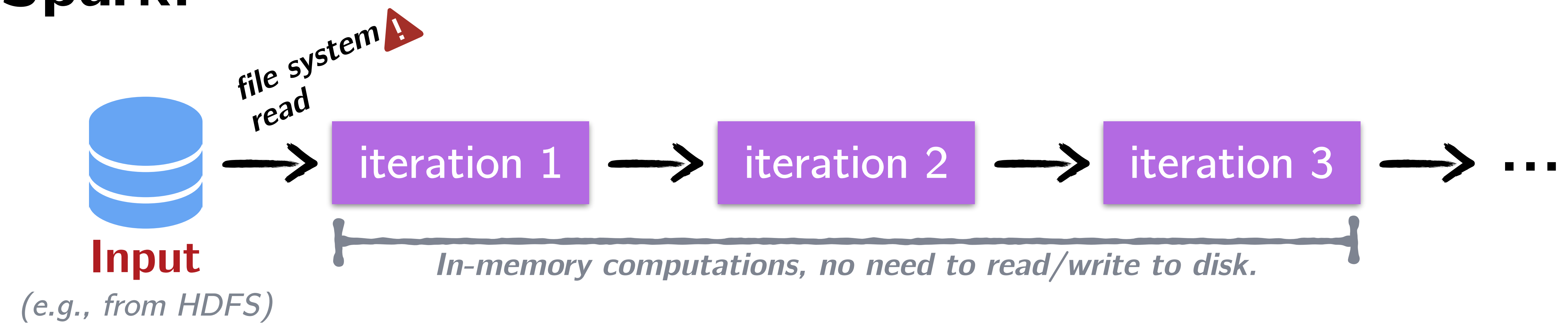
>90% of time in IO that Spark can avoid.

Iteration in Hadoop:



Input
(e.g., from HDFS)
file system read ⚠
iteration 1
MapReduce program
file system write ⚠
Read/write intermediate data
file system read ⚠
iteration 2
MapReduce program
file system write ⚠
Read/write intermediate data
file system read ⚠
iteration 3
MapReduce program
file system write ⚠
Read/write intermediate data
…

Iteration in Spark:

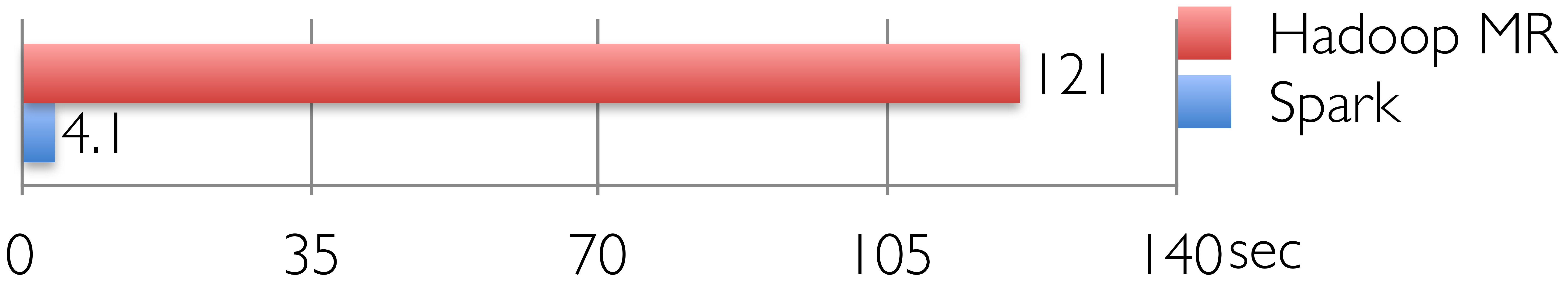


Input
(e.g., from HDFS)
file system read ⚠
iteration 1
iteration 2
iteration 3
…
In-memory computations, no need to read/write to disk.

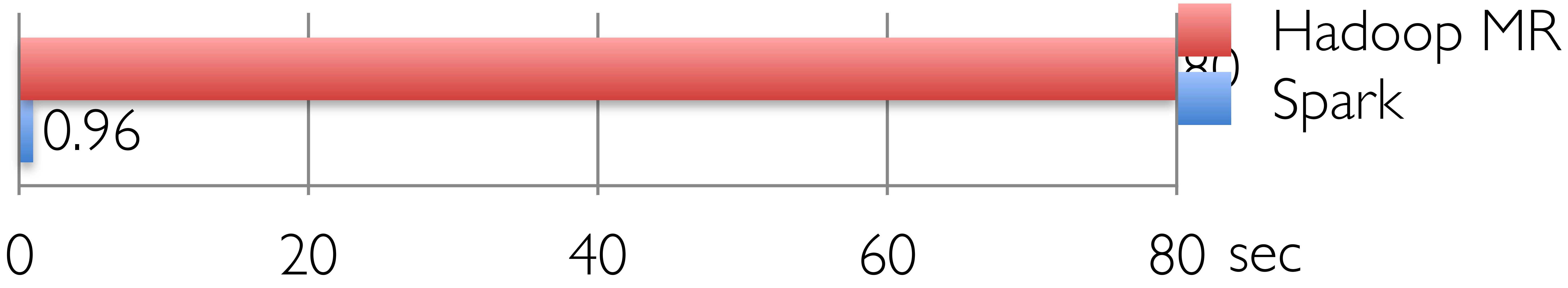
In-Memory Can Make a Big Difference

(2013) Two iterative Machine Learning algorithms:

K-means Clustering



Logistic Regression



INTRO TO SPARK

Three APIs

WE'LL FOCUS ON THESE:

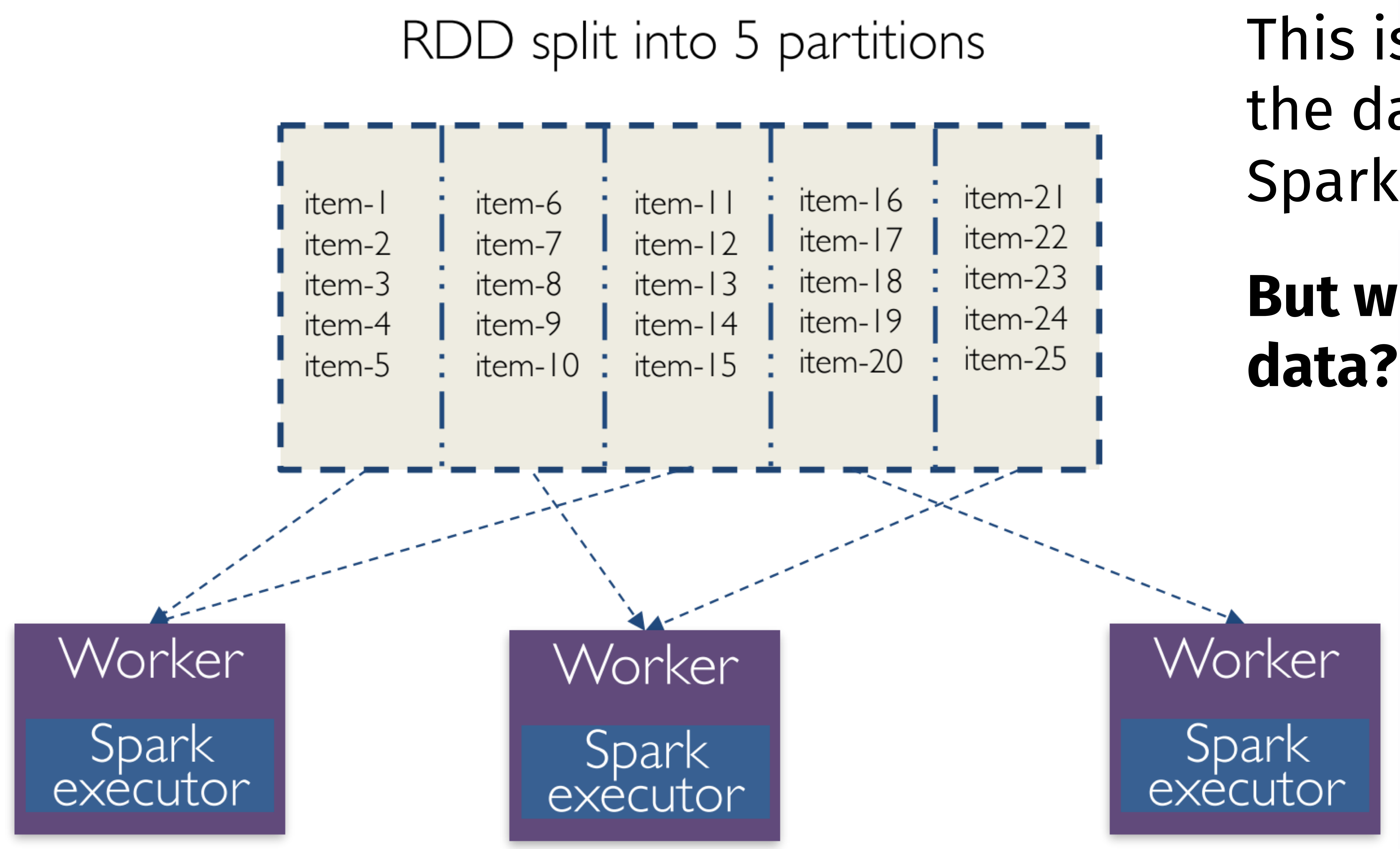
1. **RDDs.** (Resilient Distributed Datasets.)
Think: distributed list.

2. **DataFrames**. SQL-like structured datasets with query operations.

3. **Datasets**. A mixture of RDDs and DataFrames in terms of the operations that are available on them.

SPARK'S CORE DATA REPRESENTATION

RDDs (Resilient Distributed Datasets)


RDD split into 5 partitions
item-1
item-2
item-3
item-4
item-5
item-6
item-7
item-8
item-9
item-10
item-11
item-12
item-13
item-14
item-15
item-16
item-17
item-18
item-19
item-20
item-21
item-22
item-23
item-24
item-25
Worker
Spark executor
Worker
Spark executor
Worker
Spark executor

This is how you can think about the data being laid out over Spark's architecture.

But what about operations on this data?

SPARK'S CORE DATA REPRESENTATION

Operations on RDDs

There are two main classes of operations on Spark's data abstractions:

> *This is the **functional** difference you should remember!*

Transformations. Return new RDDs as a result.

Actions. Compute a result based on an RDD, and either returned or saved to an external storage system (e.g., HDFS).

SPARK'S CORE DATA REPRESENTATION

Operations on RDDs

There are two main classes of operations on Spark's data abstractions:

*This is the **semantic** difference you should remember!*

Transformations. Return new RDDs as a result.

‼ They are **lazy**, their result RDD is not immediately computed.

Actions. Compute a result based on an RDD, and either returned or saved to an external storage system (e.g., HDFS).

‼ They are **eager**, their result is immediately computed.

SPARK'S CORE DATA REPRESENTATION

Operations on RDDs

KEY POINT!:

Laziness/Eagerness is how we can limit network communication using the programming model.

two main classes of operations on Spark's data

Transformations. Return new RDDs as a result.
They are **lazy**, their result RDD is not immediately computed.

Actions. Compute a result based on an RDD, and either returned or saved to an external storage system (e.g., HDFS).
They are **eager**, their result is immediately computed.

Carnegie Mellon University
School of Computer Science

More on Transformations

Transformations create a new RDD from an existing one.
Remember a few slides back? We can tell something is a transformation because it returns a new RDD as a result.

Transformations use lazy evaluation:
Results are not computed right away – instead, Spark remembers set of transformations applied to base data set.

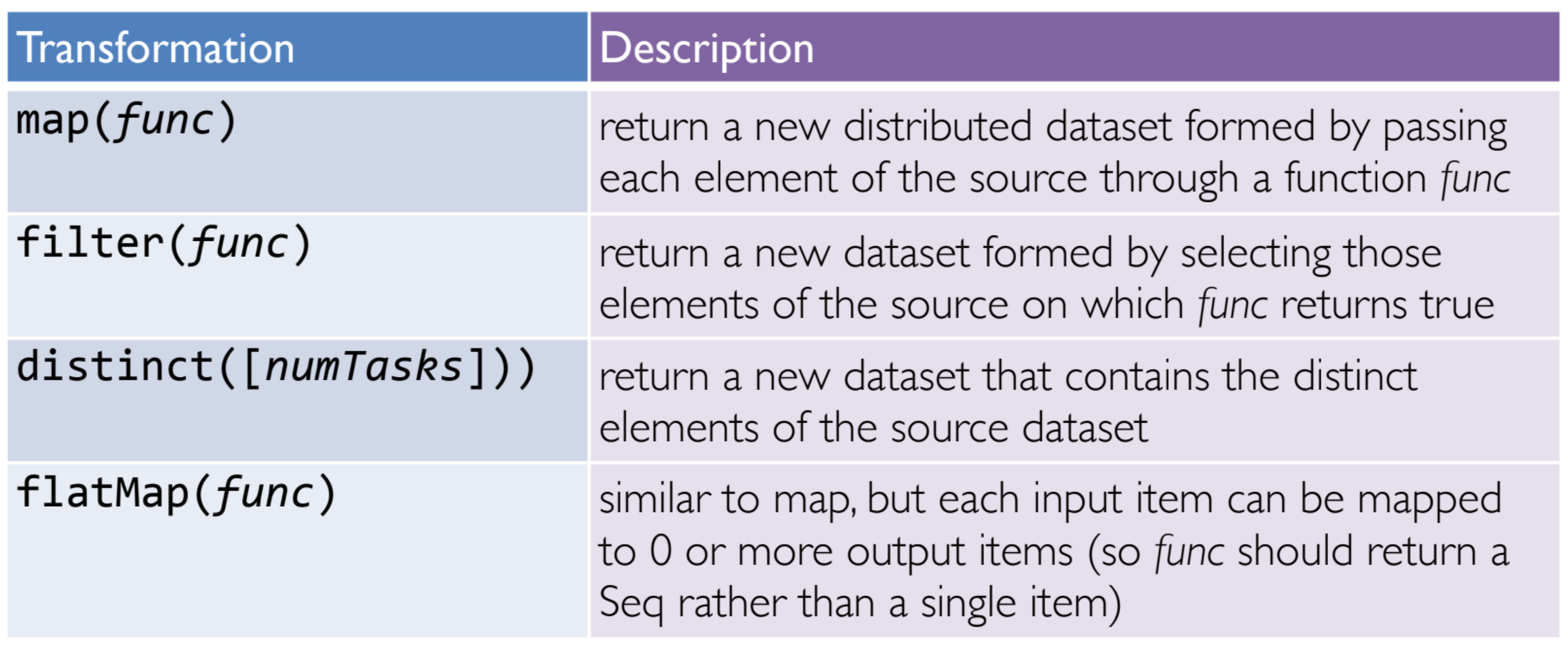
This is good because... ?

- it enables Spark to optimize the required operations
- Spark can recover from failures and slow workers

Another way to think about it...
You can think of transformations providing a recipe for computing a result.

SPARK'S CORE DATA REPRESENTATION

Some Transformations



Transformation	Description
`map(func)`	return a new distributed dataset formed by passing each element of the source through a function *func*
`filter(func)`	return a new dataset formed by selecting those elements of the source on which *func* returns true
`distinct([numTasks]))`	return a new dataset that contains the distinct elements of the source dataset
`flatMap(func)`	similar to map, but each input item can be mapped to 0 or more output items (so *func* should return a Seq rather than a single item)

More on Actions

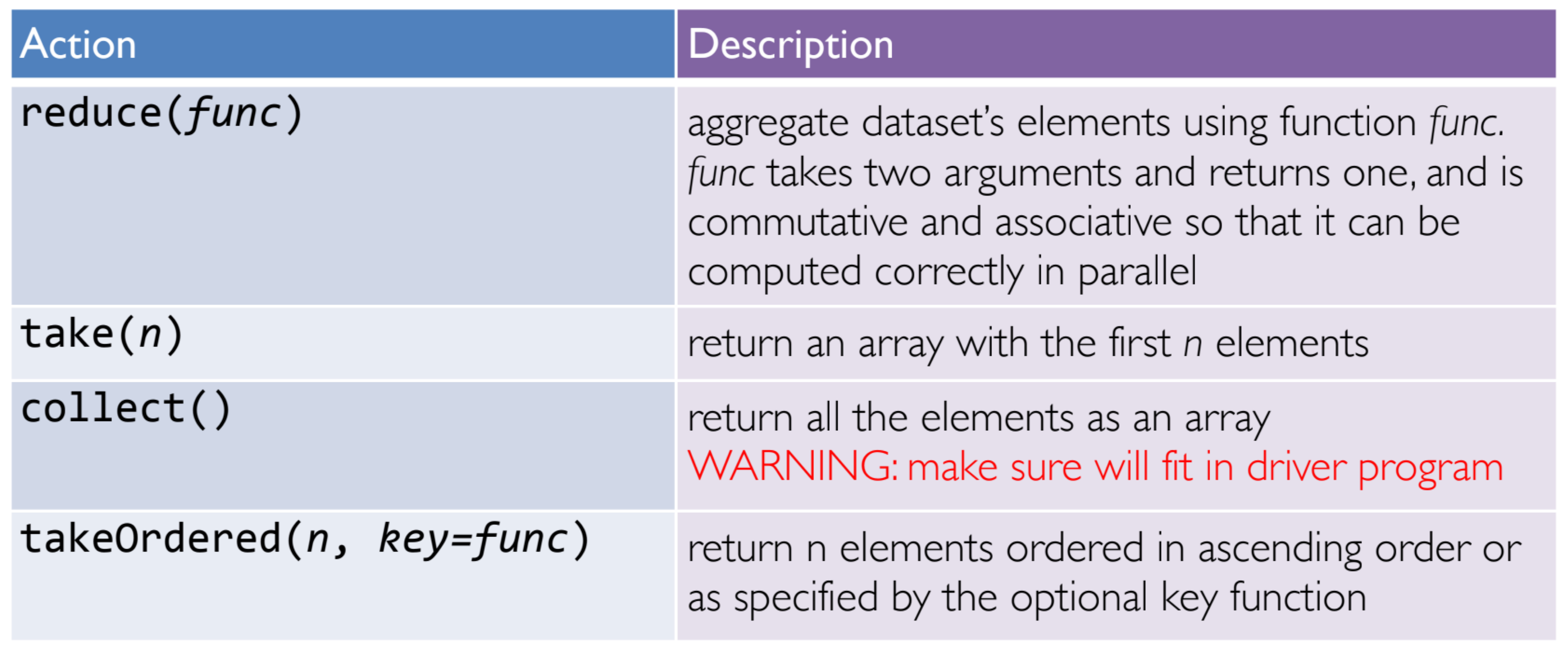
Actions cause Spark to execute recipe to transform source.
Remember a few slides back? We can tell something is an action because it returns something other than a new RDD as a result.

Actions are eager!

Actions are how we get the results of our computations out of Spark.

SPARK'S CORE DATA REPRESENTATION

Some Actions



Action	Description
`reduce(func)`	aggregate dataset's elements using function *func*. *func* takes two arguments and returns one, and is commutative and associative so that it can be computed correctly in parallel
`take(n)`	return an array with the first *n* elements
`collect()`	return all the elements as an array WARNING: make sure will fit in driver program
`takeOrdered(n, key=func)`	return n elements ordered in ascending order or as specified by the optional key function

…But how do I actually use Spark?

Recitation and first HW assignment will provide more details…

Architecture of a Spark Job
Lambda functions
RDDs and Caching
DataFrames
Examples
Using Databricks to work on HW
Submitting HW assignments to Gradescope

Outline

1. Big Data: Challenges and Opportunities
2. MapReduce / Hadoop
3. Datacenter Communication Hierarchy
4. Spark and ML
5. HW1 preview

Homework 1

- **Programming**:
  - Spark “hello, world”
  - Data cleaning: entity resolution
- **Written**:
  - PCA, nonlinear dimensionality reduction

Note: Please get started on HW1 right away!

- Last Friday’s recitation went over how to get set up with Spark/Databricks
- Written portion is meant to help test/gauge math background (optimization, combinatorics, linear algebra). Although the problems discuss dimensionality reduction, **you don’t need to know details about PCA/Laplacian Eigenmaps to answer the questions.**

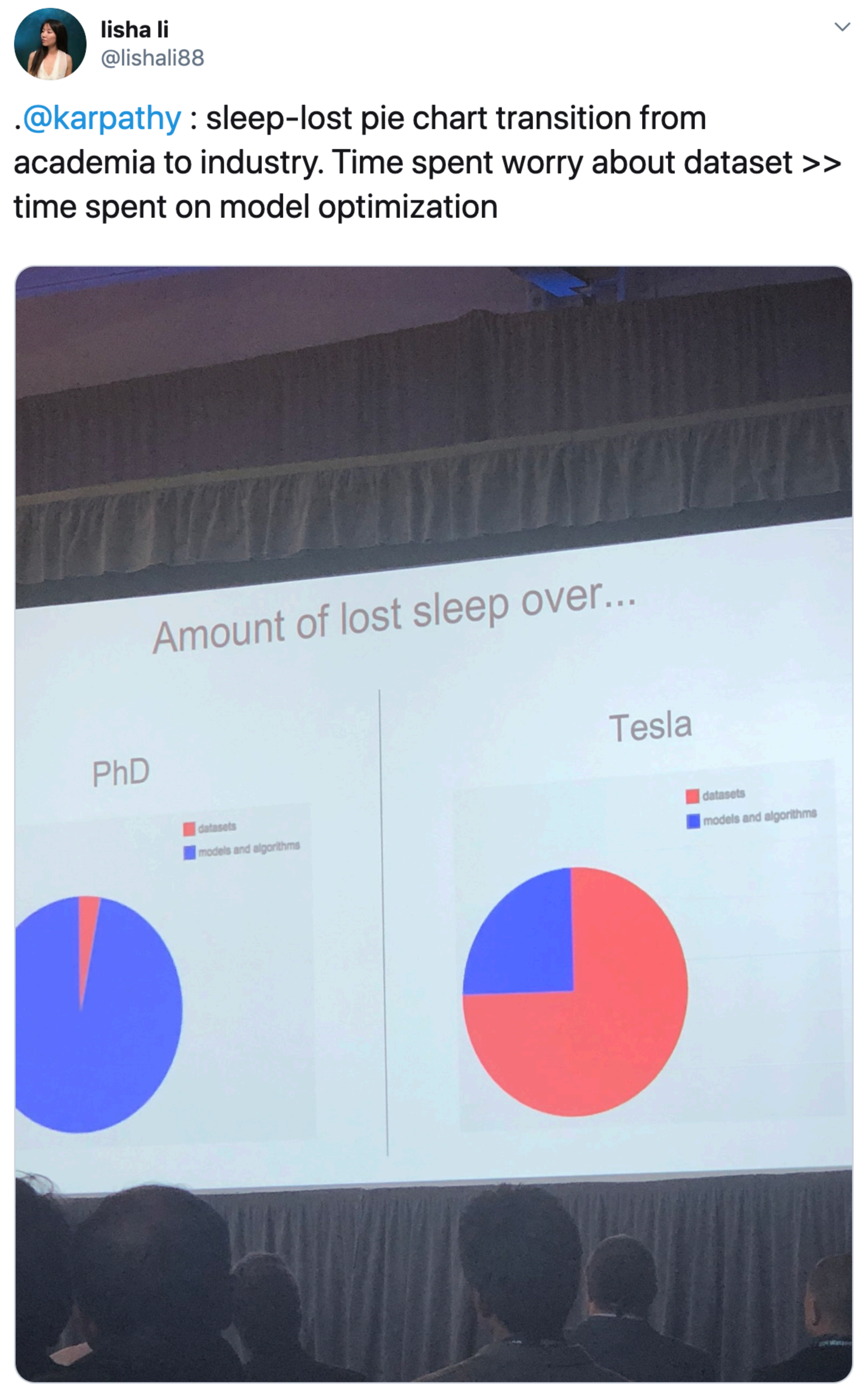
HOMEWORK 1

Data cleaning

Better data → Better ML

A critical part of the ML pipeline

- incomplete data / NaN values
- corrupted data
- incorrectly labeled data
- standardizing data
- duplicated data



lisha li
@lishali88

.@karpathy : sleep-lost pie chart transition from academia to industry. Time spent worry about dataset >> time spent on model optimization

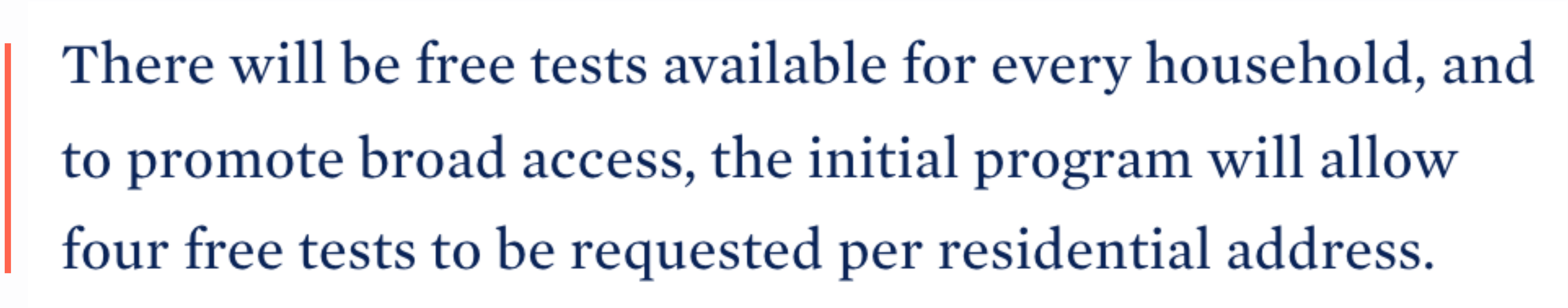
Amount of lost sleep over...
PhD
datasets
models and algorithms
Tesla
datasets
models and algorithms

HOMEWORK 1

Entity resolution example

One example of a (massive!) entity resolution problem:

https://www.whitehouse.gov/briefing-room/statements-releases/2022/01/14/fact-sheet-the-biden-administration-to-begin-distributing-at-home-rapid-covid-19-tests-to-americans-for-free/



> There will be free tests available for every household, and to promote broad access, the initial program will allow four free tests to be requested per residential address.

Name	Address
RAY RUTH	305 K ST WASHINGTON DC 20001
RAY K RUTH JR	305 K STREET NW WASHINGTON DC 20001
DIVYA GUPTA	305 K ST NW APT A WASHINGTON DC 20001-4512
REUBEN ROBINSON	305 K STREET APT A WASHINGTON, D.C., 20001

HOMEWORK 1

What is Entity Resolution?

Problem of identifying and linking/grouping different manifestations of the same real world object.

Examples of manifestations and objects:

- Different ways of addressing (names, email addresses, Facebook accounts) the same person in text.
- Web pages with differing descriptions of the same business.
- Different photos of the same object.
- ...

HOMEWORK 1

Entity Resolution Motivating Examples

- Linking Census Records
- Public Health
- Web search
- Comparison shopping
- Counter-terrorism
- Spam detection
- Machine Reading
- …

HOMEWORK 1

Entity Resolution Lab

Web scrape of Google Shopping and Amazon product listings

Google listing:

» clickart 950000 - premier image pack (dvd-rom) massive collection of images & fonts for all your design needs ondvd-rom!product informationinspire your creativity and perfect any creative project with thousands ofworld-class images in virtually every style. plus clickart 950000 makes iteasy for ...

Amazon listing:

» clickart 950 000 - premier image pack (dvd-rom)

Visually, we see these listings are the same product

Goal: algorithmically decide this.

Model and Algorithm

Model ER as Text Similarity

- » We will use a weighted bag-of-words comparison
  - Some tokens (words) are more important than others
  - Sum up the weights of tokens in each document

How to assign weights to tokens?

- » Term-Frequency/Inverse-Document-Frequency or **TF-IDF** for short

TF-IDF (Intuition)

Term-Frequency

- Rewards tokens that appear many times in the same document
  = (# times token appears)/(total # of tokens in doc)

Inverse-Document-Frequency

- rewards tokens that are rare overall in a dataset
  = (total # of docs)/(#of docs containing token)

Total TF-IDF value for a token

- Product of its TF and IDF values for each doc
- Need to remove stopwords (a, the, …) from docs

Cosine Similarity

Compare two docs by computing cosine of angle between vectors

Small angle (large cosine) means docs share many tokens in common

Large angle (small cosine) means they have few words in common

RECALL

Announcements

- **HW1**: due *Monday January 29th* at 11:59pm ET
  - please start early!
  - use Piazza, office hours, recitation for questions
    - but don’t provide direct answers on these forums

- **Questions about Spark/Databricks?**
  - Watch recitation from Friday
  - Ask on Piazza